A STUDY OF THE METHOD OF LOCAL VARIATIONS FOR SOLVING OPTIMAL CONTROL PROBLEMS

By
CHANDRA SEKHAR SIRCAR

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

OCTOBER, 1973

A STUDY OF THE METHOD OF LOCAL VARIATIONS FOR SOLVING OPTIMAL CONTROL PROBLEMS

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By
CHANDRA SEKHAR SIRCAR

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
OCTOBER, 1973

CERTIFICATE

This is to certify that the thesis entitled ''A Study of the Method of Local Variations for Solving Optimal Control Problems'' is a record of the work carried out under my supervision and that it has not been submitted elsewhere for a degree.

S. S. Prabhu

S.S. Prabhu
Assistant Professor
Department of Electrical Engg.
Indian Institute of Technology
Kanpur

Kanpur
October, 1973

ACKNOWLEDGMENT

I would like to express my sincerest appreciation and gratitude to my thesis supervisor, Dr. S.S. Prabhu, for his constructive criticisms, invaluable suggestions and sustained interest in my work.

I am grateful to the staff of the Computer Centre for their help which aided me immensely to compile this work.

Finally, I would like to thank all my instructors and colleagues for the very pleasant time I had during my stay here.

Kanpur
October 24, 1973.

C.S. Sircar

ABSTRACT

This thesis deals with the study and some applications of the Method of Local Variations which is a neighbourhood method utilising a direct search technique for obtaining locally optimal solutions to optimal control problems. A brief review of direct and indirect methods for solving optimal control problems has been given, together with an introduction to the Method of Local Variations technique and the motivation for its study. This is followed by a detailed description of the local variations algorithm, its comparison with dynamic programming techniques, and some of its advantages and disadvantages. Next, the Method of Local Variations and some modifications on it have been applied to solve several optimal control problems. Consequently, some important features of the local variations algorithm have emerged, providing a basis for its comparison with the other better known techniques of optimisation and in drawing conclusions about its relative merits and demerits. Finally, the conclusions arrived at, together with some suggestions regarding modifications to this method for improvement of the basic algorithm, have been given.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

A problem which seeks to extremise a function (or functional) of one or more variables and in which the solution is required to satisfy certain specified constraints, may be referred to as an optimisation problem. A dynamical system may have such a problem defined for it; in the field of control theory, this problem is known as an optimal control problem.

The formulation of an optimal control problem requires:

a) a mathematical model of the process to be controlled,

b) a statement of the physical constraints and

c) the specification of a performance criterion.

Attention will be confined, in this study, to dynamical systems which can be described by state equations of the form:

$$\dot{\underline{x}}(t) = \underline{a}(\underline{x}(t), \underline{u}(t), t) \qquad (1)$$

where $\underline{x}(t)$ is the n-dimensional state vector,

$\underline{u}(t)$ is the m-dimensional control vector and

$\underline{a}$ is an n-dimensional vector function which is, in general, nonlinear and time-varying, with arguments as indicated.

Thus, the systems considered are, in general, finite-dimensional, deterministic, continuous-time and nonlinear-time-varying.

Considering an optimal control problem defined in the finite time interval $[t_o, t_f]$ where t_o is the initial time and t_f the final time, the constraints may consist of the specified boundary conditions on the state variables together with any state and control constraints that may be imposed in the interval $[t_o, t_f]$.

The performance index is a mathematical expression which, when extremised, indicates that the system is performing in the most desirable manner. Its choice depends on the type of problem being considered. We will consider performance indices which are expressible in the form:

$$J = h(\underline{x}(t_f), t_f) + \int_{t_o}^{t_f} g(\underline{x}(t), \underline{u}(t), t)dt \qquad (2)$$

where h and g are scalar functions of the arguments indicated in equation (2).

A typical optimal control problem can, thus, be formulated as follows: It is required to find an admissible control $\underline{u}^*$ which causes the system described by equation (1) to follow an admissible trajectory $\underline{x}^*$ that minimises the performance index given by equation(2), subject to the constraints stipulated. $\underline{u}^*$ is called an

optimal control and $\underline{x}^*$ an optimal trajectory. The problems considered in this thesis all belong to the above category.

There are several methods of solution of such problems. These methods can be broadly classified into two main categories, namely, direct methods and indirect methods.

1.1 Direct and Indirect Methods for Solving Optimal Control Problems

Indirect methods usually utilise the necessary conditions for optimality derived from the Calculus of Variations or Pontryagin's Minimum Principle. Solution of optimisation problems by indirect methods can be obtained in the following way:

a) a set of necessary conditions that the optimum must satisfy are derived;

b) the equations constituting the necessary conditions are then solved to obtain candidates for the optimum and

c) the candidates for the optimum are tested by using the necessary- and-sufficient-condition tests.

A feature of indirect methods is that state and control constraints increase the complexity involved in obtaining optimal solutions.

Application of the Calculus of Variations or Pontryagin's Minimum Principle to optimal control problems

gives rise to 'Two Point Boundary Value Problems' (TPBVP), and the difficulties associated with obtaining solution in this manner are due to the difficulty involved in solving complex TPBVP's defined for simultaneous, nonlinear-time-varying differential equations. This aspect has been discussed, for example, by Athans [1], Kirk [2], and Sage [3].

There are many techniques for solving TPBVP's encountered in the above formulation. The quasilinearisation technique is one such method for finding a solution to these TPBVP's. Athans [1] has given a brief description of this technique while a more elaborate treatment can be found in Sage [3]. The quasilinearisation technique has been applied by Bellman and Kalaba [5,6,7,8], McGill and Kenneth [9], Kopp and McGill [10], and Moyer and Pinkham [11] for solving TPBVP's to obtain optimal control. This technique has also been applied by Kumar and Sridhar [12], Detchmendy and Sridhar [13], and Sage and Eisenberg [14] to solve several engineering problems in optimal control theory. Application of this method to discrete systems has been investigated by Henrici [15], Sylvester and Meyer [16], Sage and Burt [17], and Sage and Smith [18].

The invariant imbedding procedure is another method for solving TPBVP's. It is a technique whereby the missing initial (or terminal) conditions are

directly obtained. An extensive treatment of this method appears in Sage [3] while Bellman et al.[19], Detchmendy and Sridhar [20], Sage and Masters [21,22], and Sage and Ellis [23,24] have applied this technique to solve several optimal control problems.

There are many other methods for solving these TPBVP's. Some well known methods are the Steepest Descent Method, the Method of Variation of Extremals, the Conjugate Gradient Method, and the Second Variation Method. Kirk [2] has given a detailed description of the Steepest Descent Method and the Method of Variation of Extremals. Lasdon et al. [25] have described the Conjugate Gradient Method while a discussion on Second Variation Methods appears in Bryson and Ho [26].

An important feature of indirect methods is that a solution obtained, utilising the necessary conditions for optimality, may not be an optimal solution at all. Even if it is optimal, since the necessary conditions obtained from the Calculus of Variations or Pontryagin's Minimum Principle are for local optima only, the result will, in general, hold only locally.

Direct methods are those which do not utilise the necessary conditions for optimality to obtain a solution to the optimal control problem. Pierre [4] defines direct methods of solution as those which require the direct use of the performance index and the constraint

equations of a given problem, while systematic recursive methods are employed to obtain an optimal solution. An important feature of many direct methods is that an initial feasible trajectory forms the starting point in the algorithm which finally converges to the desired optimal solution. At any iteration stage, the nominal trajectory is a feasible one. Thus, the formulation helps us in picking out a good suboptimal trajectory with less effort than is required to find an optimal trajectory. Moreover, since most of the direct methods are search techniques, any state or control inequality constraint that may be specified, helps in narrowing down the domain of search and eases the finding of a solution to the problem being considered. These methods were primarily devised to circumvent the problems associated with obtaining solutions to TPBVP's.

An important example of a direct method is Bellman's Dynamic Programming technique [27,28,29]. This is a sophisticated search technique for solving multi-stage decision making problems. This technique is, in fact, the repeated, sequential, stage-by-stage application of the optimality principle of Bellman. It can be applied to continuous time processes as well as to discrete time ones. Although this technique provides a global optimal control law, it suffers from a serious malady known as the 'curse of dimensionality' in that the memory requirement is very

large for any but very low order problems. To overcome this serious drawback, several modifications have been made on the original algorithm giving rise to such techniques as Incremental Dynamic Programming of Bernholtz and Graham [30], Differential Dynamic Programming of Jacobson and Mayne [31], and State Increment Dynamic Programming of Larson [32]. All these techniques achieve a reduction in the memory requirment at the cost of losing the globality of the optimal control law.

Another example of a direct method for solving optimal control problems is the Gradient Projection Method of Rosen [33,34,35]. It is an iterative numerical procedure for finding an extremum of a function of several variables that are required to satisfy various constraints. A detailed description of this method in its application to optimal control problems appears in Kirk [2].

1.2 Motivation for the Study of the Method of Local Variations

Incremental Dynamic Programming of Bernholtz and Graham [30], Differential Dynamic Programming of Jacobson and Mayne [31] and State Increment Dynamic Programming of Larson [32] are all modified versions of Bellman's Dynamic Programming technique and come under the category of neighbourhood techniques using direct methods of optimisation. All these techniques have one aim in

common - a reduction in computer memory requirement. They achieve this at the cost of losing the globality of the optimal control law that they provide.

In the incremental dynamic programming technique as applied to a single state variable case, a neighbourhood is defined about the nominal state trajectory being considered. This trajectory consists of a discrete set of state points at certain instants of time. The dynamic programming algorithm is applied, in this neighbourhood only, to these state points, starting from the last state and sweeping backwards, such that the best trajectory, in that neighbourhood, is generated. This newly generated trajectory now forms the current nominal trajectory around which a neighbourhood is again defined and the whole process is repeated. This goes on till a locally optimal trajectory is obtained. For the vector case, the neighbourhood is defined considering perturbations in one state variable at a time, and the best trajectory, in that neighbourhood, is generated for perturbations in that state variable only, keeping all other state variables the same. Once this trajectory has been generated for that particular state variable, the next state variable is then similarly treated. This process is repeated till all the state variables have been considered. This whole procedure is repeated till satisfactory convergence to an optimal control is obtained.

Differential dynamic programming is a successive approximation technique based on Bellman's dynamic programming method. This technique is used for determining optimal control for nonlinear systems. It uses a linearised version of the Hamilton-Jacobi-Bellman Equation (HJBE) at each of the time instants into which the time interval, for the problem being considered, is divided. The optimal control satisfies the HJBE. It is assumed that the optimal control is unknown in the time interval being considered, but that a nominal control is available in the same time period. On application of this nominal control a nominal state trajectory, in the same time interval, is produced by equation (1). The nominal performance index value can be calculated from equation (2). It is assumed that the optimal performance index expression is sufficiently well-behaved to allow a power series expansion in a small neighbourhood around the nominal state trajectory. In each iteration, the system equations are integrated in forward time, using the current nominal control. The accessory equations, which yield the coefficients of a linear or quadratic expansion of the performance index expression in the neighbourhood of the nominal state trajectory, are integrated in reverse time, thus yielding an improved control law because the performance index, which is a function of both state and control

Differential dynamic programming is a successive approximation technique based on Bellman's dynamic programming method. This technique is used for determining optimal control for nonlinear systems. It uses a linearised version of the Hamilton-Jacobi-Bellman Equation (HJBE) at each of the time instants into which the time interval, for the problem being considered, is divided. The optimal control satisfies the HJBE. It is assumed that the optimal control is unknown in the time interval being considered, but that a nominal control is available in the same time period. On application of this nominal control a nominal state trajectory, in the same time interval, is produced by equation (1). The nominal performance index value can be calculated from equation (2). It is assumed that the optimal performance index expression is sufficiently well-behaved to allow a power series expansion in a small neighbourhood around the nominal state trajectory. In each iteration, the system equations are integrated in forward time, using the current nominal control. The accessory equations, which yield the coefficients of a linear or quadratic expansion of the performance index expression in the neighbourhood of the nominal state trajectory, are integrated in reverse time, thus yielding an improved control law because the performance index, which is a function of both state and control

variables, has been approximated within a small neighbourhood of the nominal state trajectory which was, in turn, produced by a nominal control. This improved control law is applied to the system equations producing a new and improved trajectory. By continued iterations, this procedure produces control functions that successively approximate the optimal control function. These successive approximation algorithms have been found to converge rapidly, for a large class of problems, to a locally minimal solution. Memory requirements of these algorithms are modest when compared with conventional dynamic programming. The sacrifice relative to conventional dynamic programming is, of course, global optimality.

Another attempt to overcome the dimensionality barrier set up by conventional dynamic programming is the method of state increment dynamic programming. The essential difference between this technique and the conventional method lies in the choice of the time interval over which a given control is applied. In conventional dynamic programming, the sampling interval Δt is fixed, whereas in the state increment dynamic programming, the time increment δt is the minimum time required for at least one of the state variables to change by one increment. Thus, the next state, for a given control, is known to lie within some small neighbourhood of the point at which the control is

applied. This procedure computes the optimal control in blocks which may cover a long time interval but only a small distance along each state variable.

The method of local variations is yet another example of a neighbourhood technique using a direct method of optimisation. This method has features of both the calculus of variations and dynamic programming. Chernous'Ko [36], and Krylov and Chernous'Ko [37] have given a detailed description of the method and its applications. Prakasa Rao et al. [38] have applied this method to solve an optimal scheduling problem in hydro-thermal power systems. It has been found that this method may yield a local optimal solution and that its memory requirement is less than that required for incremental dynamic programming [38]. Banichuk et al.[39] have successfully applied the method of local variations to solve two point boundary value problems also.

The salient features of this method are a discrete-time description of the system whose state equations are generally differential, an initial feasible trajectory as a starting point of the solution, and a systematic, iterative perturbation, in a specified small neighbourhood of the current feasible trajectory, such that the performance index value monotonically decreases from iteration to iteration till a satisfactory convergence to an optimal trajectory is obtained. At

every stage of the algorithm, all the constraints of the problem are satisfied. In general, in each iteration, a sequence of mathematical programming problems need to be solved. This aspect has been discussed in more detail in Chapter 2, Section 2.4.

This method does not seem to have been applied widely for obtaining numerical solutions to optimal control problems. There is, thus, a need to study this method in its application to optimal control problems with a view to gain computational experience and insight into the working of this method, as well as to see how it compares with the more widely used techniques of optimisation.

CHAPTER 2

THE METHOD OF LOCAL VARIATIONS

In the previous chapter, the Method of Local Variations (MLV) has been introduced as a neighbourhood technique using a direct method of optimisation. It has also been mentioned that this method exhibits features of both the Calculus of Variations and Dynamic Programming, and may provide locally optimal solutions to optimal control problems. In this chapter, a detailed description of the method will be given. This will be followed by its comparison with some Dynamic Programming techniques together with a list, with explanations, of the advantages and disadvantages of MLV.

2.1 The Elementary Operation

In order to facilitate understanding of the MLV algorithm, we will give a definition of an Elementary operation. We will follow the approach of Krylov and Chernous'Ko [37] by formulating a typical optimal control problem to be solved by the MLV technique.

Let us suppose that the process to be controlled is described by the

State Equation: $\dot{\underline{x}} = \underline{f}(\underline{x},\underline{u},t)$ (2.1-1)

where $\underline{x}$ is the n-dimensional state vector,

$\underline{u}$ is the m-dimensional control vector,

$\underline{f}$ is an n-dimensional vector function which is, in general, nonlinear and time-varying, with arguments as indicated, and

t is the independent variable.

The process is assumed to be described in the finite time interval [o,T] which is considered to be fixed.

It is required to minimise the performance index

$$J = \int_o^T g\,(\underline{x},\underline{u},t)\,dt \qquad (2.1\text{-}2)$$

where g is a scalar function of the arguments indicated in equation (2.1-2).

subject to

State Constraint: $\underline{x}(t) \in G(t)$ for $o \leq t \leq T$

where G(t) is a variable closed region of the n-dimensional Cartesian space $\mathbb{R}^n$

and

Control Constraint: $\underline{u}(t) \in U(t,\underline{x})$ for $o \leq t \leq T$

where $U(t,\underline{x})$ is a variable closed region of the m-dimensional Cartesian space $\mathbb{R}^m$

under the

Initial Condition: $\underline{x}(o) \in G(o)$

and

Final Condition: $\underline{x}(T) \in G(T)$

Let t_1 and t_2 be two sufficiently close instants of time, and $\underline{x}_1$ and $\underline{x}_2$ two sufficiently close state points

in the phase space, such that $o \leq t_1 \leq t_2 \leq T$, $\underline{x}_1 \in G(t_1)$ and $\underline{x}_2 \in G(t_2)$. Let us consider the problem of determining a control $\underline{u}(t)$ in the time interval $[t_1, t_2]$, which translates the system given by equation (2.1-1) from co-ordinates $(t_1, \underline{x}_1)$ to co-ordinates $(t_2, \underline{x}_2)$ and minimises the incremental performance index

$$\Delta J = \int_{t_1}^{t_2} g(\underline{x}, \underline{u}, t)\, dt \qquad (2.1\text{-}3)$$

subject to the state and control constraints already stipulated being satisfied in the time interval $[t_1, t_2]$. This problem will be referred to as an Elementary Problem, and the process of determining a solution to this problem as an Elementary Operation.

Considering the pair of near-points $(t_1, \underline{x}_1)$ and $(t_2, \underline{x}_2)$, an elementary operation must provide:

a) an answer as to whether there exists a control which translates the system given by equation (2.1-1) from co-ordinates $(t_1, \underline{x}_1)$ to co-ordinates $(t_2, \underline{x}_2)$ and satisfies the stipulated constraints ;

b) if yes, an optimal control $\underline{u}^*(t)$ which solves the above variational problem, and the corresponding minimum incremental performance index value ΔJ^*. ~~For any pair of sufficiently~~ For any pair of sufficiently near-points $(t_1, \underline{x}_1)$ and $(t_2, \underline{x}_2)$, an elementary operation can be carried out approximately. Due to the proximity of $(t_1, \underline{x}_1)$

and $(t_2,\underline{x}_2)$ and the possibility of performing the elementary operation approximately, the elementary problem becomes, in a large number of cases, an algebraic problem only. In addition, the nearness of $(t_1,\underline{x}_1)$ and $(t_2,\underline{x}_2)$ can also be used for simplifying state equation (2.1-1) by linearising it. The set $G(t)$ is usually a region of the n-dimensional state space changing continuously with time. As the ends of the trajectory, $(t_1,\underline{x}_1)$, $(t_2,\underline{x}_2)$, are sufficiently near and lie in this region, the state constraint set can be approximated by a linear inter polation between $G(t_1)$ and $G(t_2)$ in $[t_1,t_2]$. The set $U(t,\underline{x})$ can be considered as approximately fixed, in carrying out the elementary operation at

$$\tilde{U}(t,\underline{x}) \approx \tilde{U}\left[(t_1+t_2)/2,\ (\underline{x}_1+\underline{x}_2)/2\right]$$

2.2 A Description of the MLV Algorithm

It is assumed that the elementary operation, satisfying the conditions formulated above, has been constructed for the problem under consideration. This elementary operation forms the basis of the Method of Local Variations algorithm; by the repeated, sequential application of this operation, it may be possible to obtain a locally optimal solution to the optimisation problem as described below.

In order to apply the MLV algorithm, the state equation and the performance index expression, given by

equations (2.1-1) and (2.1-2) respectively, are discretised. This is done by dividing the time interval [o,T] into N equal subintervals to obtain a discrete-time representation of the system at time instants $t_k=k\tau$; (k=o,1,...,N), where $\tau=T/N$ is sufficiently small. Next, an initial feasible trajectory $\underline{x}^o(t_k)$; (k=o,1,...,N), is chosen such that the state and control constraints stipulated, including the boundary conditions on the state, are satisfied. The performance index value corresponding to this trajectory is calculated.

This initial trajectory now becomes the current nominal trajectory at the zeroeth iteration. Every state point on this trajectory is given a sequential perturbation of +h or -h according to the MLV procedure described below, giving rise to a sequence of elementary problems which may be solved by performing a sequence of elementary operations. After all the state points in the current nominal trajectory have been perturbed and the resulting elementary problems have been solved by elementary operations, one iteration is complete and a state trajectory which is better than the previous nominal trajectory, together with the corresponding control sequence and performance index value, become available for the next iteration. This newly generated state trajectory now becomes the current nominal trajectory and the MLV perturbation scheme is applied to it. This procedure is

repeated until a specified stopping criterion is satisfied. Thus, at every iteration, a feasible trajectory, together with the corresponding control sequence and performance index value, are available.

The variation of the current nominal trajectory $\underline{x}(t_k)$; (k=o,1,...,N), by the method of local variations consists of perturbing the state at each value of k by an n-dimensional perturbation vector $\underline{h}$, and, if the perturbed state is admissible,solving the resulting elementary problem. This procedure is applied at time instants t_k; (k=o,1,...,N), sequentially. The approach suggested in Krylov and Chernous' Ko [37] for the perturbation scheme for elementary operations is the following:

Each of the n components of $\underline{x}(t_k)$; (k=o,1,...,N), are successively perturbed in a particular order - say the first component, then the second and so on. For perturbation given to each component of the state at time instant t_k, the associated elementary problems are solved, before proceeding to perturb the next component of the state, at the same time instant. For each of the components x_j; (j = 1,...,n), its own perturbation step $h_j > o$, is given. Let the current nominal trajectory in the pth iteration, $\underline{x}^p(t_k)$; (k=o,1,,..,N), be varied by the MLV technique, as described below, to obtain the nominal trajectory for the (p+1)th iteration, $\underline{x}^{p+1}(t_k)$;

$(k=o,1,\ldots,N)$. Let it be assumed that the variation of the state values at the time instant t_ℓ is under consideration. Then variation of state values at time instants t_k; $(k=o,1,\ldots,\ell-1)$, have already been performed. So the current nominal trajectory in the pth iteration at this sta e is:

$\tilde{\underline{x}}^p(t_k)$; $(k=o,1,\ldots,\ell-1)$, $\underline{x}^p(t_k)$; $(k=\ell,\ldots,N)$, where

$\underline{x}^p(t_k)$;$(k=o,1,\ldots,\ell-1)$, represents the new values assigned to the nominal state trajectory in the pth iteration by the application of the MLV technique at time instants t_k; $(k=o,1,\ldots,\ell-1)$. Each component of $\underline{x}^p(t_\ell)$, starting from the first component and going upto the nth, is varied as follows, assuming, for simplicity, that all the components of the n-dimensional perturbation vector $\underline{h}$ are equal to h :

A perturbation +h is given to the component under consideration. The following steps are then taken:

a) A test is performed to see whether this perturbed value of the component under consideration of the state $\underline{x}^p(t_\ell)$ is admissible; if admissible we go to step b) below, otherwise this step is repeated with a perturbation of -h to the same component; in the event of both perturbations leading to an inadmissible state value, we go to step d) below;

b) the optimisation problem of determining admissible control inputs in the time intervals $[t_{\ell-1}, t_\ell]$, $[t_\ell, t_{\ell+1}]$

which take the system from the state $\tilde{\underline{x}}^p(t_{\ell-1})$ at time instant $t_{\ell-1}$ to the state $\underline{x}^p(t_{\ell+1})$ at time instant $t_{\ell+1}$ via the perturbed state at time instant t_ℓ, is then solved. If there does not exist a solution to this problem, the perturbation given to $\underline{x}^p(t_\ell)$ is considered a 'failure' and the above steps are repeated for perturbation in the same component of $\underline{x}^p(t_\ell)$ of -h; if we meet with this 'failure' for -h perturbation also, we go to step d) below;

c) the incremental performance index values for the above state transfer in the time intervals $[t_{\ell-1}, t_\ell]$, $[t_\ell, t_{\ell+1}]$ are calculated. If the sum of these two incremental performance index values is less than the corresponding sum that exists before the above perturbation scheme for the state component under consideration, the perturbation is considered to be a 'success' and the perturbed state component value is assigned to the nominal trajectory at the time instant t_ℓ and the whole perturbation scheme is then applied to the next component of the state at time t_ℓ.

d) If perturbations of both +h and -h meet with 'failures', the unperturbed value of the state component at time instant t_ℓ is retained and the above steps are repeated for perturbations of the next state component at the same time instant t_ℓ, till the state components have been subjected to the above operations.

After all the n components of the state $\underline{x}^p(t_\ell)$ have been varied by the above procedure, the algorithm moves on to the state point $\underline{x}^p(t_{\ell+1})$ and varies the n components of that state point in the same manner. In this way, the local variations operation is performed successively for all n components of all the state points, upto time instant t_N. This completes the pth iteration, and the nominal state trajectory and control sequence, together with the modified performance index value, are obtained for the (p+1)th iteration. The variation of the initial and final points in the state trajectory differs in the fact that the elementary operations are carried out only " to the right" or "to the left" respectively. Thus, each iteration generates a feasible state trajectory (alongwith the corresponding control sequence and performance index value) which is better than (or at least as good as) the state trajectory in the previous iteration, in the sense of the performance index. These iterations, which successively reduce the performance index value, are performed till some convergence criterion is satisfied, thereby producing a locally optimal solution to the problem being considered.

Figure 2.2 explains the working of the local variations scheme for a scalar case. It shows a part of the nominal trajectory at any iteration and considers state values at

time instants (I-1), I and (I+1). A perturbation +h has been given to the state value at the Ith time instant.

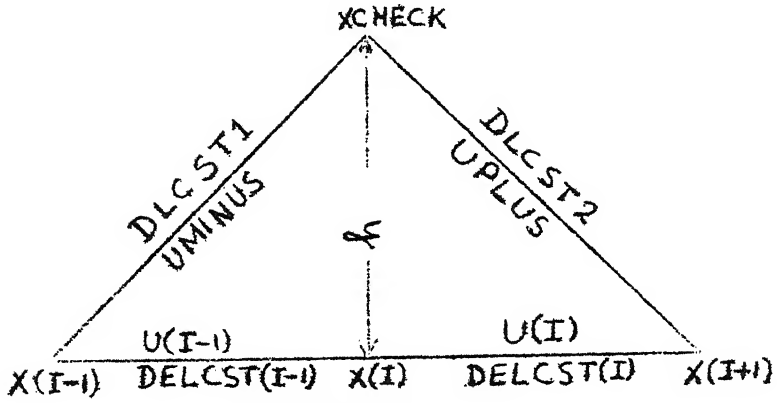


Figure 2.2

XCHECK is the state value at the Ith time instant, perturbed by +h.

U(I-1) is the nominal control input in the time interval [I-1,I].

U(I) is the nominal control input in the time interval [I,I+1).

DELCST(I-1) is the incremental performance index value associated with the transfer of X(I-1) to X(I).

DELCST(I) is the incremental performance index value associated with the transfer of X(I) to X(I+1).

UMINUS is the modified control input required to transfer X(I-1) to XCHECK.

UPLUS is the modified control input required to transfer XCHECK to X(I+1).

DLCST1 is the modified incremental performance index value associated with the transfer of X(I-1) to XCHECK.

DLCST2 is the modified incremental performance index value associated with the transfer of XCHECK to X(I+1).

The perturbation given to X(I) poses an elementary problem. An elementary operation tries to solve this problem by finding out the following:

a) whether the perturbed state XCHECK is admissible;

b) if yes, whether there exist control sequences UMINUS and UPLUS, capable of transferring X(I-1) to X(I+1) via XCHECk; if so, whether UMINUS and UPLUS are admissible; if yes, whether they are unique; if not, we are faced with a mathematical programming problem whose solution provides the optimal control inputs UMINUS and UPLUS;

c) assuming UMINUS and UPLUS exist, are admissible and unique, whether the sum of incremental performance index values (DLCST1+DLCST2) associated with the transfer of X(I-1) to X(I+1) via XCHECK, is less than the corresponding sum [DELCST(I-1)+DELCST(I)] associated with the transfer of X(I-1) to X(I+1) via X(I).

If the answers to a), b) and c) above are <u>all</u> in the affirmative, the perturbation +h to X(I) is a 'success'. Otherwise, it is a 'failure' and a perturbation -h is now given to X(I). This situation can similarly be analysed as above.

2.3 <u>Comparison of the Method of Local Variations with Dynamic Programming Techniques</u>

Bellman's Dynamic Programming [27,28,29] is a direct computational procedure for solving multi-stage decision

making problems by the repeated, sequential, stage-by-stage application of the principle of optimality. This technique provides a global optimal control law in a closed-loop form. It is applicable to both linear and nonlinear systems. As it is a direct search technique, any state or control constraints that may be stipulated in the problem, help to reduce the domain of search and ease the finding of a solution. The major limitation of dynamic programming is the storage requirement in digital computation. Bellman refers to this difficulty as the 'curse of dimensionality'. This feature is the penalty that must be paid for avoiding the TPBVP's that arise in the case of variational methods. Furthermore, as it is a systematic, exhaustive search procedure for multi-stage processes, the amount of computations may become prohibitively large for all but very low order systems.

To avoid the 'curse of dimensionality', several modifications were done on the original dynamic programming algorithm. This resulted in the development of such techniques as Incremental Dynamic Programming [30], Differential Dynamic Programming [31] and State Increment Dynamic Programming [31], all of them being neighbourhood techniques using a direct method of optimisation. The Method of Local Variations is yet another neighbourhood technique utilising a direct method of optimisation and requiring significantly less storage capacity than the conventional dynamic programming technique.

Here also, any state or control constraints present in the problem help in finding a solution. The MLV technique may provide a local optimal control law and has been shown by Prakasa Rao et al.[38] to require a significantly less storage capacity than the incremental dynamic programming technique.

Prakasa Rao et al.[38] have compared Incremental Dynamic Programming (IDP) with the Method of Local Variations in terms of computational requirements. Both IDP and MLV are methods involving direct search in a neighbourhood of the nominal trajectory. In IDP, the principle of optimality is applied to obtain the best trajectory in this neighbourhood and the iterative procedure indicated in Section 1.2 enables a locally optimal trajectory to be obtained. On the other hand, the MLV tries to obtain, in each iteration, only a better trajectory than the existing nominal trajectory, in the neighbourhood under consideration. The incremental dynamic programming algorithm was suggested by Bernholtz and Graham [30] in the context of optimal scheduling of hydro-thermal power generating systems. For the sake of comparison, Prakasa Rao et al.[38] considered a power system consisting of one hydro plant and one thermal plant. Let the interval of optimisation [o,T] be divided into N equal subintervals, each of duration $\Delta t=1$.

The hydro-system is described by the

State Equation $x(k+1) = x(k)+L(k)-u(k)-e(k);(k=1,...,N)$ (2.3-1)

where $x(k)$ is the storage capacity of the hydro reservoir at the beginning of the kth time period,

$L(k)$ is the inflow rate into the reservoir during the kth time period,

$u(k)$ is the discharge rate from the reservoir during the kth time period and

$e(k)$ is the evaporation rate from the reservoir during the kth time period.

It is required to find the discharge rate $u^*(k)$ such that the fuel cost in thermal power generation over the interval $[o,T]$, given by

$$J = \sum_{k=1}^{N} F\,(P_T(k)) \qquad (2.3\text{-}2),$$

is minimised subject to the

Equality Constraint: $P_D(k)-P_H(k)-P_T(k)+P_{LOSS}(k)=o;(k=1,...,N)$ (2.3-3)

and the

Inequality Constraints: $x_{min} \leq x(k) \leq x_{max};(k=2,...,N)$

$u_{min} \leq u(k) \leq u_{max};(k=1,...,N)$

$P_{Tmin} \leq P_T(k) \leq P_{Tmax};(k=1,,,,,N)$

where $F(P_T(k)) = a\,P_T(k) + b\,P_T^2(k)$; $(k=1,\ldots,N)$ (2.3-4),

is the fuel cost of generating P_T thermal units during the kth time period, and a,b are constants,

$P_D(k)$ is the load demand on the system in the kth time interval,

$P_H(k)$ is the power generated in the hydro station in the kth time interval,

$P_T(k)$ is the power generated at the thermal station during the kth time interval and

$P_{LOSS}(k)$ is the power lost in transmission during the kth time interval,

The initial and final reservoir storages x(1) and x(N+1) are specified.

Prakasa Rao et al.[38] have considered a case in which the interval of optimisation for this one-hydro-one-thermal problem is divided into three equal subintervals as shown in Figure 2.3.

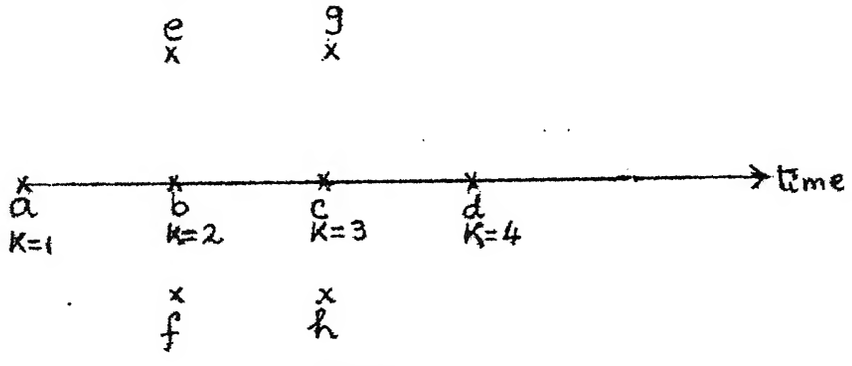


Figure 2.3

The points a,b,c,d represent the nominal state trajectory in a particular iteration. The states e,f,g,h define the neighbourhood considered in these methods. It has been shown that for IDP method, nine trajectories need to be compared to find the optimal trajectory in this neighbourhood, whereas for MLV technique, five trajectories are to be compared in the worst case, and three trajectories in the best case. The MLV technique not only requires a lesser number of trajectories for comparison than the IDP technique but also the number of main storage locations required by the MLV technique is significantly less than that required by the IDP technique. These results, obtained by Prakasa Rao et al.[38] for the above problem, have been tabulated in Tables 2.3(a) and (b); details can be found in Prakasa Rao et al. [38].

Number of Intervals	IDP		
	Trajectories to be compared	Primary storage requirement	No. of main calculations per iteration
3	9	24	9
6	243	51	24
N	3^{N-1}	3(3N-1)	(5N-6)

Table 2.3(a)

Number of Intervals	MLV				
	Trajectories to be compared		Primary Storage Requirement	No. of main calculations per iteration	
	Worst case	Best case		Worst case	Best case
3	5	3	15	8	4
6	11	6	24	20	10
N	(2N-1)	N	3(N+2)	4(N-1)	2(N-1)

Table 2.3(b)

From Tables 2.3(a) and (b) it can be concluded that for the problem under consideration, MLV requires a significantly lesser number of storage locations, lesser number of trajectories for comparison, and fewer number of calculations per iteration than IDP. Thus, for the same neighbourhood, MLV seems to be preferable to IDP as a method of optimisation. When the initial trajectory is far away from the optimal one, MLV provides a faster transition from the current nominal trajectory to the next one, as compared to IDP. If the optimal trajectory lies within the neighbourhood being considered at any stage of the iterative procedures of IDP and MLV, IDP leads to the exact optimal trajectory, whereas MLV may provide only a better trajectory than the existing one. It is thus advisable to use MLV initially, till we are close to the optimal trajectory, and then switch over to IDP for convergence to

the optimal trajectory. Alternately, the disparity between the solutions given by IDP and MLV can be reduced by choosing progressively smaller perturbation steps in MLV as the optimum is approached.

2.4 Some advantages and disadvantages of the Method of Local Variations

Listed below are some of the advantages and disadvantages of the MLV technique.

Advantages:

1) The Method of Local Variations has been found to be attractive due to its simplicity, efficiency and ease of implementation.

2) As MLV is a neighbourhood technique using a direct method of optimisation, even before convergence to an optimal has been achieved, a feasible and practically satisfactory solution may be chosen. Also, any state or control constraints present in the problem under consideration help in narrowing down the domain of search and ease the finding of a solution to that problem.

3) Inequality constraints can be directly handled without using any 'artificial' techniques like the penalty function approach.

4) This technique requires significantly less computer storage space than the conventional dynamic programming and IDP techniques because only the solution for the current approximation

has to be stored. As this approximation is improved, it can be 'erased' by replacing it by the new approximation.

5) Chernous' Ko [36] and Banichuk et al.[39] have applied the MLV algorithm to solve linear and nonlinear boundary value problems defined for simultaneous ordinary differential equations. These problems may arise in mathematical physics and optimal control theory.

6) In the original algorithm, the time interval of the problem being considered, is divided into N equal subintervals. It may be possible to gain computational advantage in the MLV technique by partitioning the time interval into unequal subintervals. Some other modifications of the basic MLV algorithm to improve its efficiency have been given in Chapters 3,4.

Disadvantages

1) The starting point in the solution to an optimal control problem by the MLV technique is an initial feasible trajectory. Generation of this trajectory may prove to be difficult, depending on the constraints on the problem being solved. Also, the solution to a problem may be sensitive to the initial feasible trajectory chosen as shown in Solution 1(b) to Problem 1 in Section 3.1.

2) There is no guarantee that the MLV technique will provide an optimal solution to a problem, as has been shown in

Solution 1(a) to Problem 1 in Section 3.1. If at all it provides an optimal solution, the results would be generally valid only locally.

3) In this technique, since the search is confined to a particular neighbourhood, we have to go from one neighbourhood to another in order to converge to a locally optimal solution. If the step size chosen is very small at the initial stage of the solution, the number of iterations required for convergence to a local optimal will be very large, making the algorithm inefficient. This can be avoided by taking a crude step size in the initial stage of the solution and gradually refining it as we approach the optimum. We can also, simultaneously, divide the problem time interval into gradually smaller subintervals and then apply the MLV algorithm.

4) In general, we may encounter a mathematical programming problem for each +h or -h perturbation. Referring to Figure 2.2 we see that with a perturbation of +h to the state value at the Ith time instant, that is, X(I), we are required to find control inputs UMINUS and UPLUS satisfying the control constraint, which will translate the state X(I-1) to the state X(I+1) via the perturbed state XCHECK which should satisfy the state constraint, such that the incremental performance index value for this path, given by (DLCST1+DLCST2), is less than that for the original path

X(I-1)$\longrightarrow$X(I)$\longrightarrow$X(I+1), given by [DELCST(I-1)+DELCST(I)] This involved sequence of mathematical programming problems in every iteration may be time consuming in certain problems.

CHAPTER 3

COMPUTATIONAL RESULTS

The algorithm of the Method of Local Variations (MLV) has been applied to obtain numerical solutions to certain optimal control problems. As a result, some important features of the algorithm have emerged.

3.1 A Linear, Constrained, Optimal Control Problem

Problem 1

State Equation: $x(k+1) = x(k) + u(k)$;

$(k = 0,1,\ldots,N-1)$ (3.1-1)

It is required to minimise the performance index

$$J = x^2(N) + \sum_{k=0}^{N-1} [u^2(k)+x^2(k)] \qquad (3.1\text{-}2)$$

subject to the following constraints:

State Constraint: $0 \leq x(k) \leq 1.5$; $(k = 0,1,\ldots,N)$

Control Constraint: $-1 \leq u(k) \leq 1$; $(k = 0,1,\ldots,N-1)$

under the following boundary conditions:

Initial Condition: $x(o) = 1.5$

Final Condition: $x(N)$ is free.

In this problem, $N = 100$.

Solution 1(a)

Since the state equation and the performance index are given as discrete equations, they can be

directly utilised in the Method of Local Variations. The first step to be taken is the generation of initial feasible state and control sequences. These are sequences which satisfy the boundary conditions as well as the stipulated state and control constraints. This problem of generating initial feasible state trajectories and the corresponding feasible control sequences may be quite complicated, depending upon the boundary conditions and the state and control constraints. In fact, this is one of the disadvantages of direct methods of optimisation like the MLV which require a starting feasible state trajectory and the corresponding control input. However, for the problem under consideration, an initial feasible state trajectory can be easily chosen. It was chosen to be:

$$x(k) = 1.5 \; ; \; (k = 0,1,\ldots,N) \qquad (3.1\text{-}3)$$

The control sequence corresponding to this state trajectory is

$$u(k) = 0 \; ; \; (k = 0,1,\ldots,N-1) \qquad (3.1\text{-}4)$$

From equation (3.1-4) it is seen that the control sequence generated from the choice of the initial feasible state trajectory, is admissible.

The initial state trajectory and control sequence were utilised by a computer program utilising the Method

of Local Variations to find the optimal solution. The main part of the program calculates the performance index value for the nominal state trajectory and control sequence and obtains an optimal control input and the corresponding state trajectory by repeatedly calling a subroutine ELEØPR. This subroutine varies the existing feasible trajectory using the MLV technique and returns a feasible state trajectory and the corresponding control input which are better than the preceding ones in the sense of the performance index. The state and control constraints are to be checked in the local variations procedure in the ELEØPR subroutine. This is done by calling a subroutine CHKPS written for this purpose. The convergence criterion chosen for this problem was that the difference between the performance indices for two successive trajectories must be less than 10^{-3}. The iterations were stopped when this criterion was satisfied.

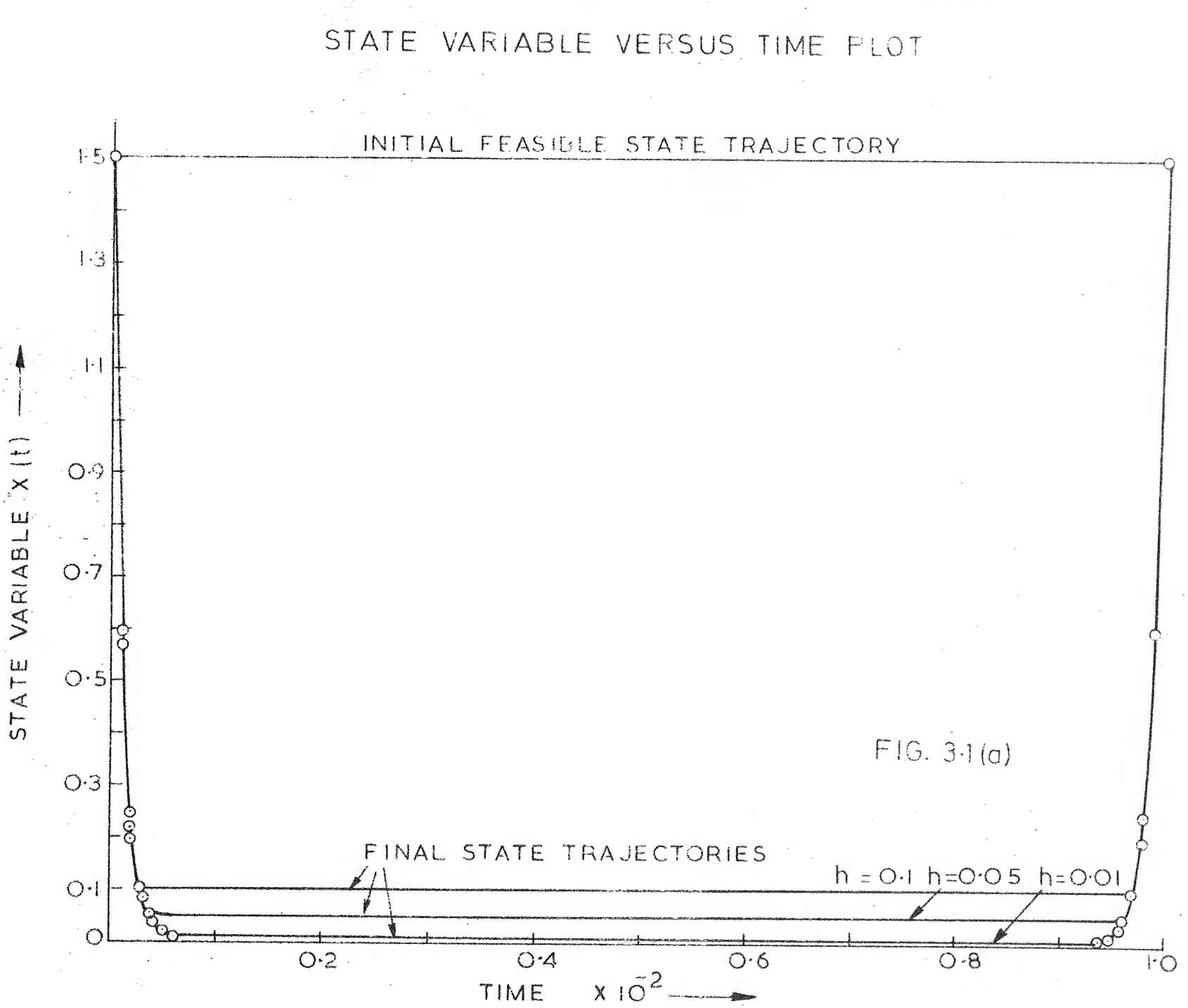
The results have been tabulated in Table 3.1(a) and the state trajectory, the control sequence, and the performance index versus iteration number plots have been shown in Figs. 3.1(a), (b) and (c) respectively.

Initial value of performance index = J_{ini}=227.250

Sl. No.	Perturbation step h	No. of iterations K	Min.value of performance index J_{min}	Execution time on IBM 7044 (secs)
1	0.1	17	5.960	5.217
2	0.05	30	5.262	9.100
3	0.01	150	5.040	25.70

Table 3.1(a)

The state trajectory shown in Fig.3.1(a) seems to be of an anomolous character because, after dropping from its initial value of 1.5 to close to zero very quickly, it again shoots up to its starting value at the final time instant, though a better result could be obtained if the control, which, starting from -1.0 and quickly going to zero as shown in the results presented, had remained zero upto the final time instant. However, the algorithm is incapable, in this particular situation, of bringing about this solution. This behaviour brings out an important feature of the MLV algorithm. For the problem and the initial state trajectory under consideration, the convergence is in the sense of the algorithm only and a claim that the converged solution is locally optimal, cannot be made.


STATE VARIABLE VERSUS TIME PLOT
INITIAL FEASIBLE STATE TRAJECTORY
STATE VARIABLE X (t)
1·5
1·3
1·1
0·9
0·7
0·5
0·3
0·1
0
0·2
0·4
0·6
0·8
1·0
TIME X 10⁻²
FINAL STATE TRAJECTORIES
h = 0·1 h=0·05 h=0·01
FIG. 3·1(a)

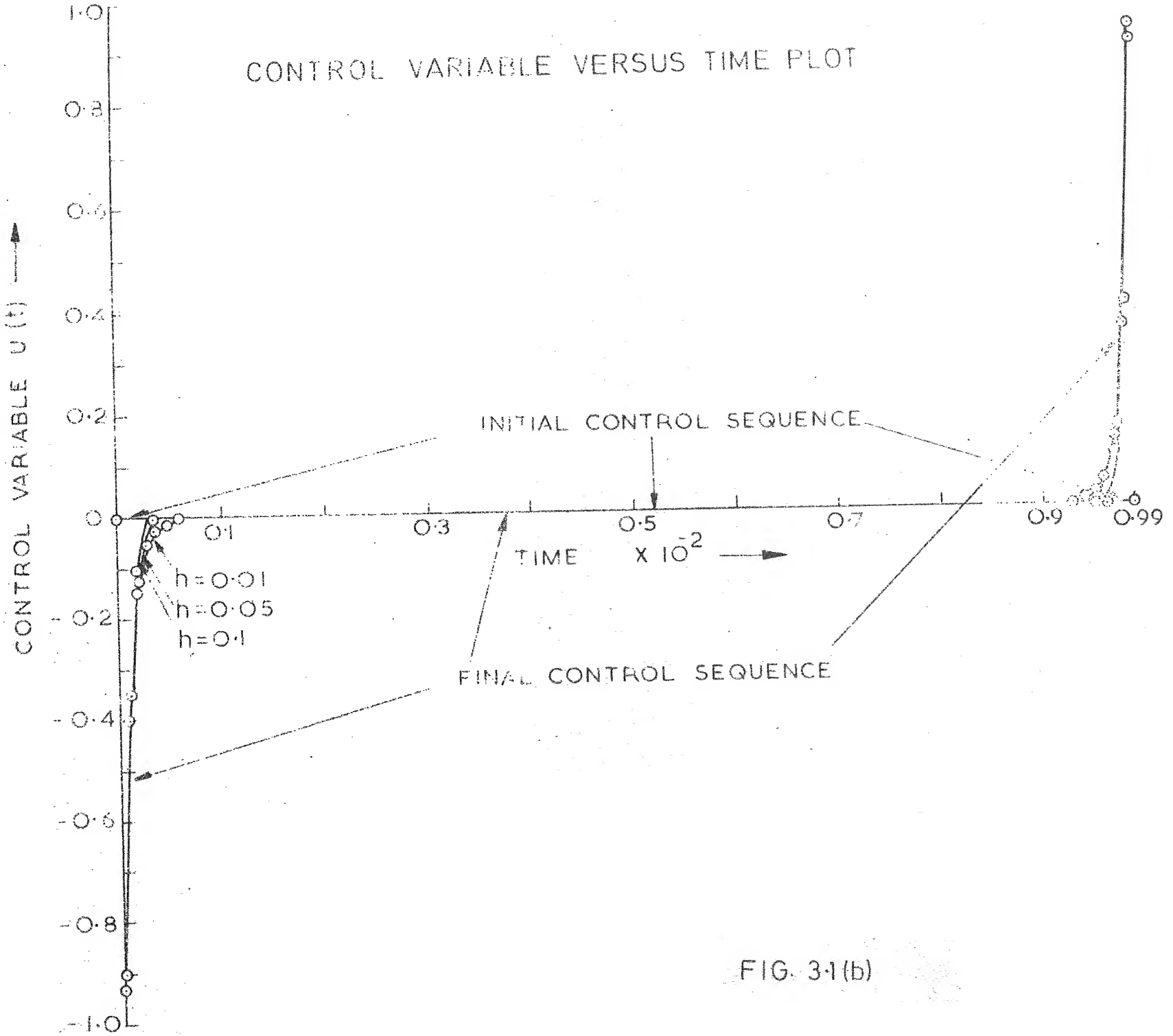

CONTROL VARIABLE VERSUS TIME PLOT
CONTROL VARIABLE U (t) ⟶
1·0
0·8
0·6
0·4
0·2
0
-0·2
-0·4
-0·6
-0·8
-1·0
0·1
0·3
0·5
0·7
0·9
0·99
TIME X 10⁻² ⟶
INITIAL CONTROL SEQUENCE
FINAL CONTROL SEQUENCE
h = 0·01
h = 0·05
h = 0·1

FIG. 3·1 (b)

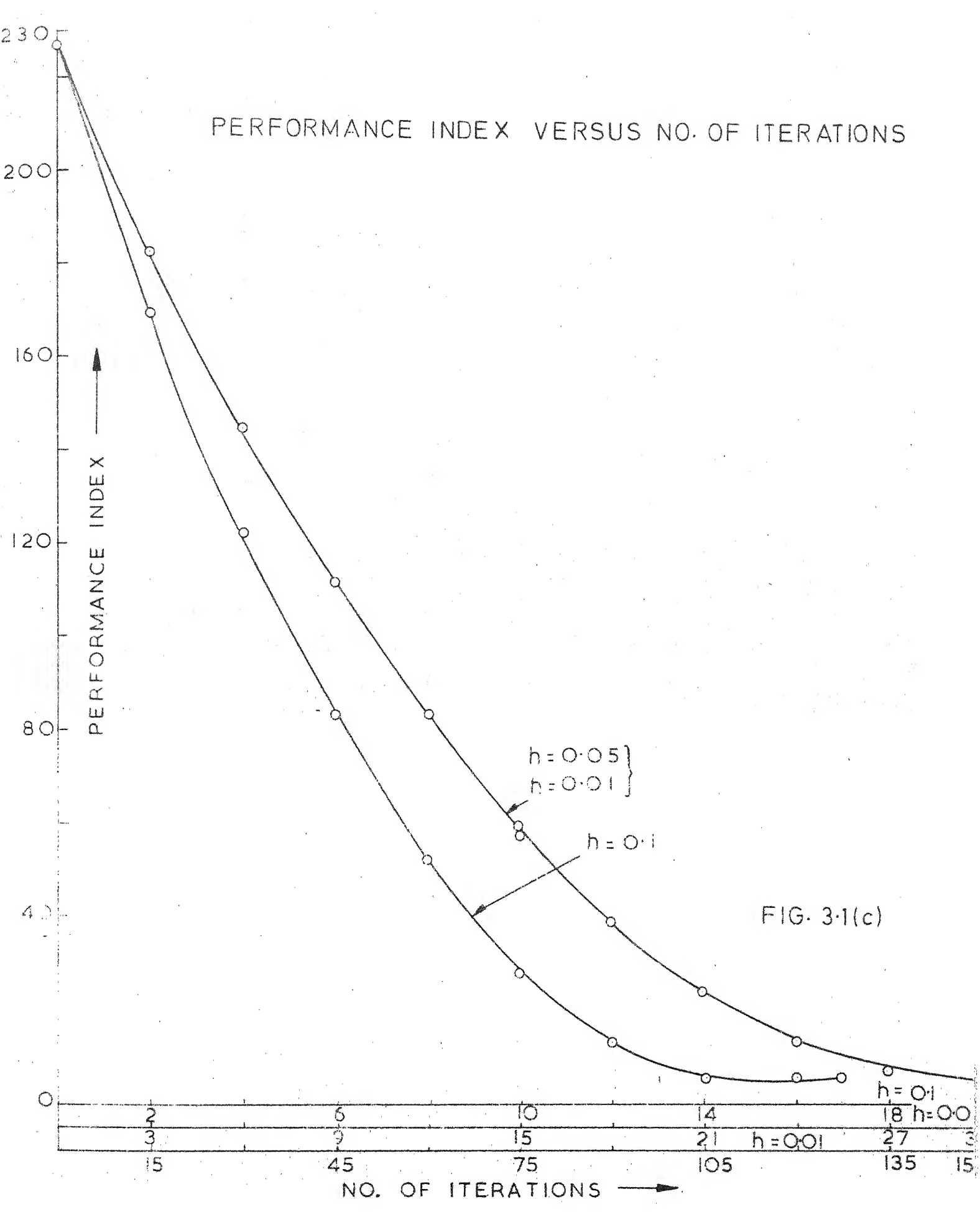

PERFORMANCE INDEX VERSUS NO. OF ITERATIONS
PERFORMANCE INDEX
230
200
160
120
80
40
0
h = 0·05
h = 0·01
h = 0·1
h = 0·1
h = 0·01
h = 0·0
2
3
6
9
10
15
14
21
18
27
15
45
75
105
135
15
NO. OF ITERATIONS

FIG. 3·1(c)

This is due to the way in which the MLV technique tries to find a better feasible trajectory in a specified neighbourhood of the current nominal trajectory - the search for a better trajectory consists in perturbing the state value at a particular time instant leaving state values at all other time instants unperturbed, starting from the time instant corresponding to N = 1. In the sense of this perturbation scheme, the algorithm fails, in the present case, to obtain a better state trajectory than the one shown in Fig.3.1(a) in its neighbourhood defined by the perturbation step size h. This, of course, does not imply that there does not exist a better trajectory in this neighbourhood - there may exist a better neighbourhood trajectory, but the algorithm may fail to find it. As a matter of fact, if the perturbation scheme of MLV is applied at N = 1,2,..., 100 to the final state trajectory shown in Fig. 3.1(a), all perturbations turn out to be failures and the trajectory of Fig.3.1(a) is retained. This cannot, strictly, be called convergence to the optimal. This important feature of the algorithm should be borne in mind when the MLV technique is used for determining optimal control.

Solution 1(b)

Now, we attempt to obtain an optimal solution to Problem 1 starting with a different initial feasible state trajectory. The initial feasible state trajectory and the corresponding control input chosen, are shown

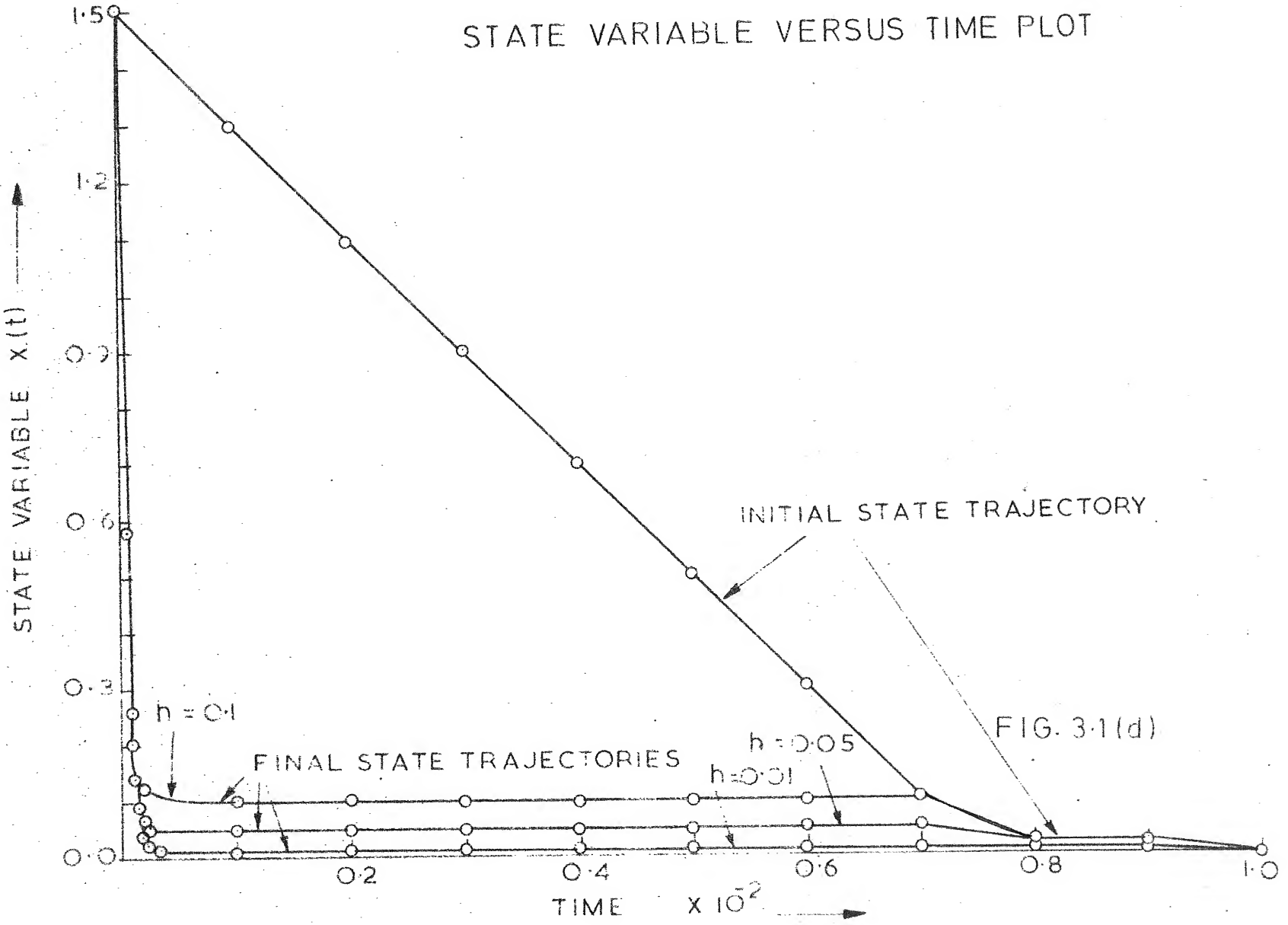
in Figures 3.1(d) and (e) respectively. The results obtained have been tabulated in Table 3.1 (b) while Figures 3.1(d), (e) and (f) show, respectively, the final state trajectory, control sequence, and the performance index plotted against iteration numbers.

Initial value of performance index = J_{ini} = 57.420

Sl. No.	Perturbation step h	No. of Iterations K	Min. value of performance index J_{min}	Execution time on IBM 7044 (secs)
1	0.1	14	4.076	3.267
2	0.05	30	3.770	6.267
3	0.01	141	3.650	27.40

Table 3.1(b)

In this problem, the final result is in accordance with our expectations and the solution obtained here is much better than Solution 1(a). The value of J_{ini} here is much less than that in Solution 1(a). The values of J_{min} here, for different values of h, are better than their counterparts in Solution 1(a). The number of iterations required here are either less (in two cases) or the same (in one case) as that in Solution 1(a). Finally, the execution times required here are less (in two cases) and more (in one case) than that in Solution 1(a). These results can be


STATE VARIABLE VERSUS TIME PLOT
STATE VARIABLE X(t)
1·5
1·2
0·9
0·6
0·3
0·0
INITIAL STATE TRAJECTORY
h = 0·1
FINAL STATE TRAJECTORIES
h = 0·05
h = 0·01
FIG. 3·1(d)
0·2
0·4
0·6
0·8
1·0
TIME X 10⁻²

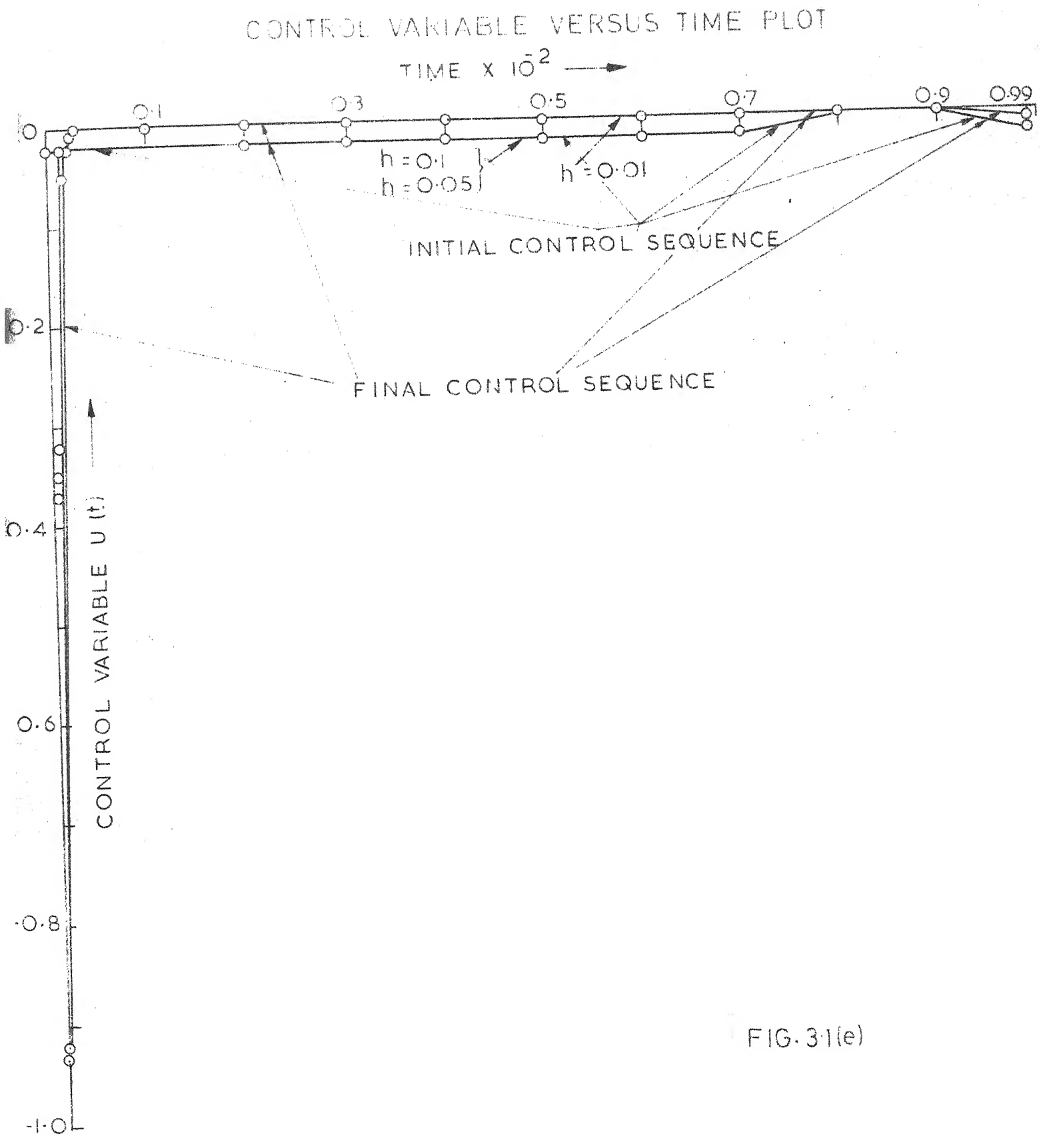

CONTROL VARIABLE VERSUS TIME PLOT
TIME X 10^{-2} ⟶
0·1
0·3
0·5
0·7
0·9
0·99
0
h = 0·1
h = 0·05
h = 0·01
INITIAL CONTROL SEQUENCE
FINAL CONTROL SEQUENCE
0·2
0·4
0·6
·0·8
-1·0
CONTROL VARIABLE U (t)

FIG. 3·1(e)

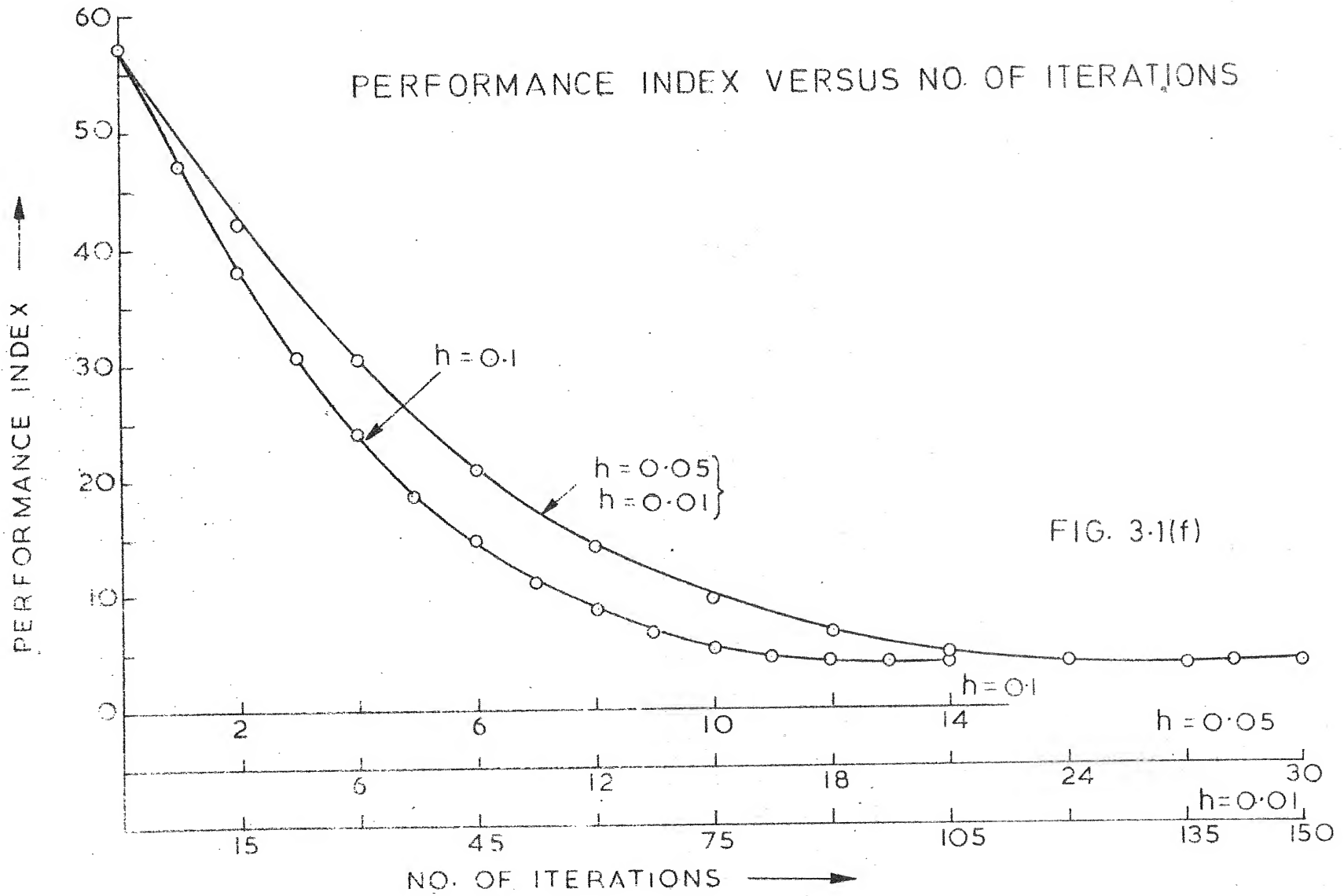

PERFORMANCE INDEX VERSUS NO. OF ITERATIONS
PERFORMANCE INDEX
60
50
40
30
20
10
0
h = 0·1
h = 0·05
h = 0·01
h = 0·1
h = 0·05
h = 0·01
2
6
10
14
30
6
12
18
24
15
45
75
105
135
150
NO. OF ITERATIONS

FIG. 3·1(f)

attributed to the fact that the choice of the initial feasible state trajectory and control sequence in Solution 1(b) were better than the earlier choice in Solution 1(a). Considering Solutions 1(a) and 1(b), it can also be observed that the result of application of MLV algorithm depends on the choice of the initial feasible state trajectory and control sequence.

It is noticed that, for both Solutions 1(a) and 1(b) to Problem 1, decrease in step size h results in rapid increase in the number of iterations required for convergence; consequently, the computation time also increases rapidly with decrease in h.

3.2 A Linear, Two-State Variable, Unconstrained, Optimal Control Problem

Problem 2

$$\dot{x}_1 = x_2 \qquad (3.2\text{-}1)$$

$$\dot{x}_2 = -x_2 + u \qquad (3.2\text{-}2)$$

It is required to minimise the performance index

$$J = \int_0^1 [x_1^2 + x_2^2 + 0.005\, u^2]\, dt \qquad (3.2\text{-}3)$$

subject to the

Initial conditions: $x_1(o) = 0$

$x_2(o) = -1$

This problem was considered by Hsieh [41] and has been solved by Lasdon et al.[25] using the Conjugate Gradient Method.

3.2.1 Solution with the original MLV Algorithm

The first step is to discretise the state equations given by (3.2-1) and (3.2-2) and the performance index given by (3.2-3). The time interval [0,1] is divided into 100 equal subintervals, each of duration $\Delta t = 0.01$. Thus, the discrete time approximation to the problem is:

State Equations

$$x_1(k+1) = 0.01\, x_2(k) + x_1(k) \quad ; \quad (k=0,1,\ldots,99) \tag{3.2-4}$$

$$x_2(k+1) = 0.99\, x_2(k) + 0.01\, u(k) \; ; \tag{3.2-5}$$

$$(k = 0,1,\ldots,99)$$

It is required to minimise

$$J = 0.01 \sum_{k=0}^{99} [x_1^2(k) + x_2^2(k) + 0.005\, u^2(k)] \tag{3.2-6}$$

subject to the

Initial Conditions: $x_1(0) = 0$

$x_2(0) = -1$

This discretised model was utilised and the MLV technique was applied to obtain a locally optimal solution. The stopping criterion used was that the ratio of the difference between two consecutive

performance index values to the larger performance index value of the two, must be less than 10^{-6}. The results obtained have been tabulated in Table 3.2(a) which also shows the results obtained by Lasdon et al.[25]. The initial feasible state trajectory and control sequence chosen, the final state trajectory and control sequence, and the performance index versus iteration number plots are given in Figures 3.2(a), (b) and (c) respectively.

Initial value of performance index $J_{ini} = 1.094$

Sl. No.	Perturbation step h	No. of Iterations K	Min. value of performance index J_{min}	Execution time on IBM 7044 (secs)	Optimum value of performance index (Lasdon et al.[25])
1	0.01	96	0.1283	12.00	
2	0.005	190	0.1004	19.00	0.07
3	0.001	1007	0.0770	84.00	

Table 3.2(a)

Since Lasdon et al.[25] have not given the computation time in their solution, we have not been able to make a comparison, in terms of computation time, of the MLV with their method for this problem.

3.2.2 Solution with a modified version of the MLV Algorithm

A modified version of the MLV algorithm was next used to solve the above problem. The motivation

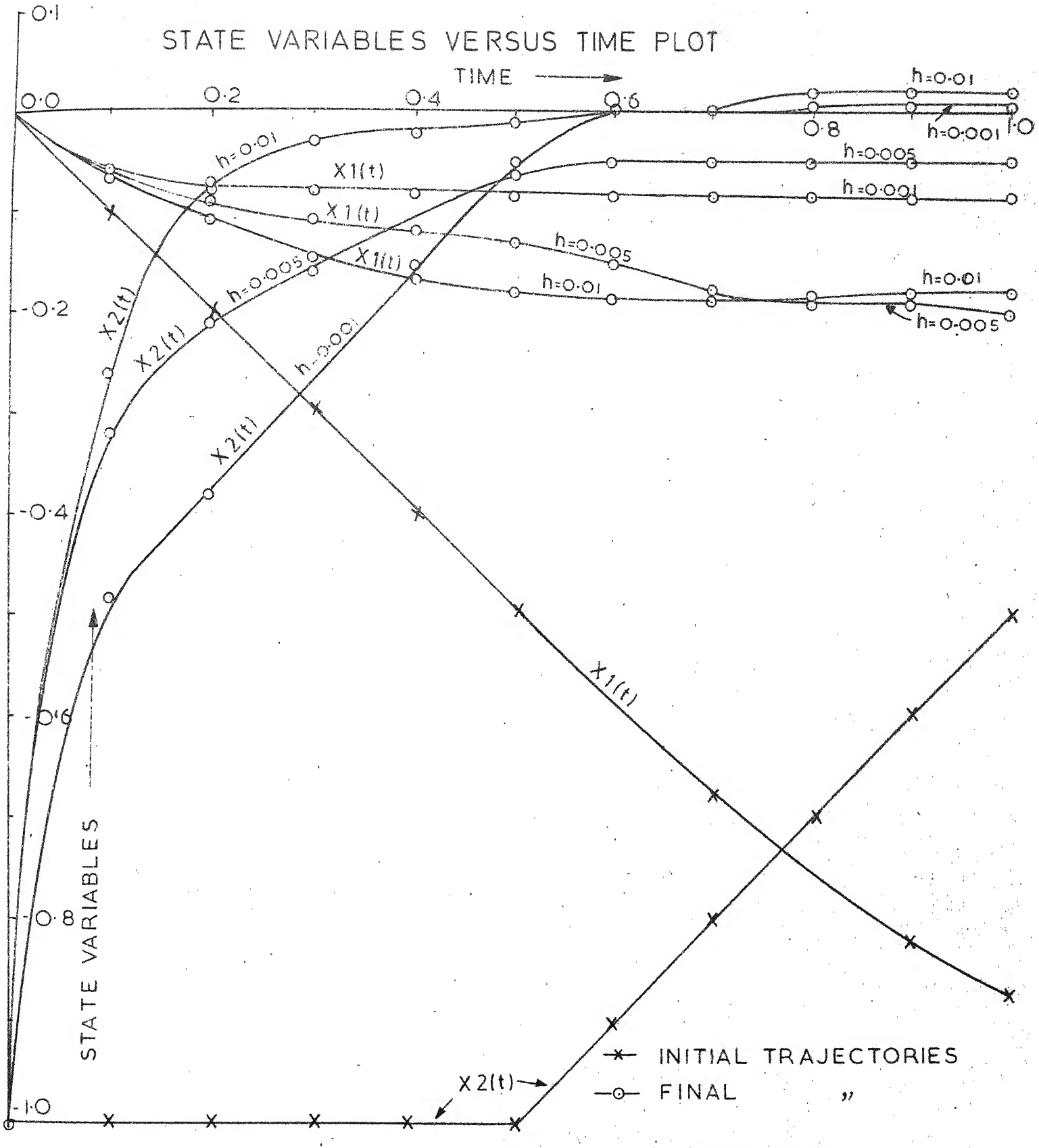

STATE VARIABLES VERSUS TIME PLOT
TIME
0·0
0·2
0·4
0·6
0·8
1·0
0·1
-0·2
-0·4
-0·6
-0·8
-1·0
STATE VARIABLES
h=0·01
h=0·001
h=0·005
h=0·001
h=0·005
h=0·01
h=0·01
h=0·005
h=0·001
X1(t)
X1(t)
X1(t)
X1(t)
X2(t)
X2(t)
X2(t)
X2(t)
INITIAL TRAJECTORIES
FINAL "

FIG. 3·2 (a)

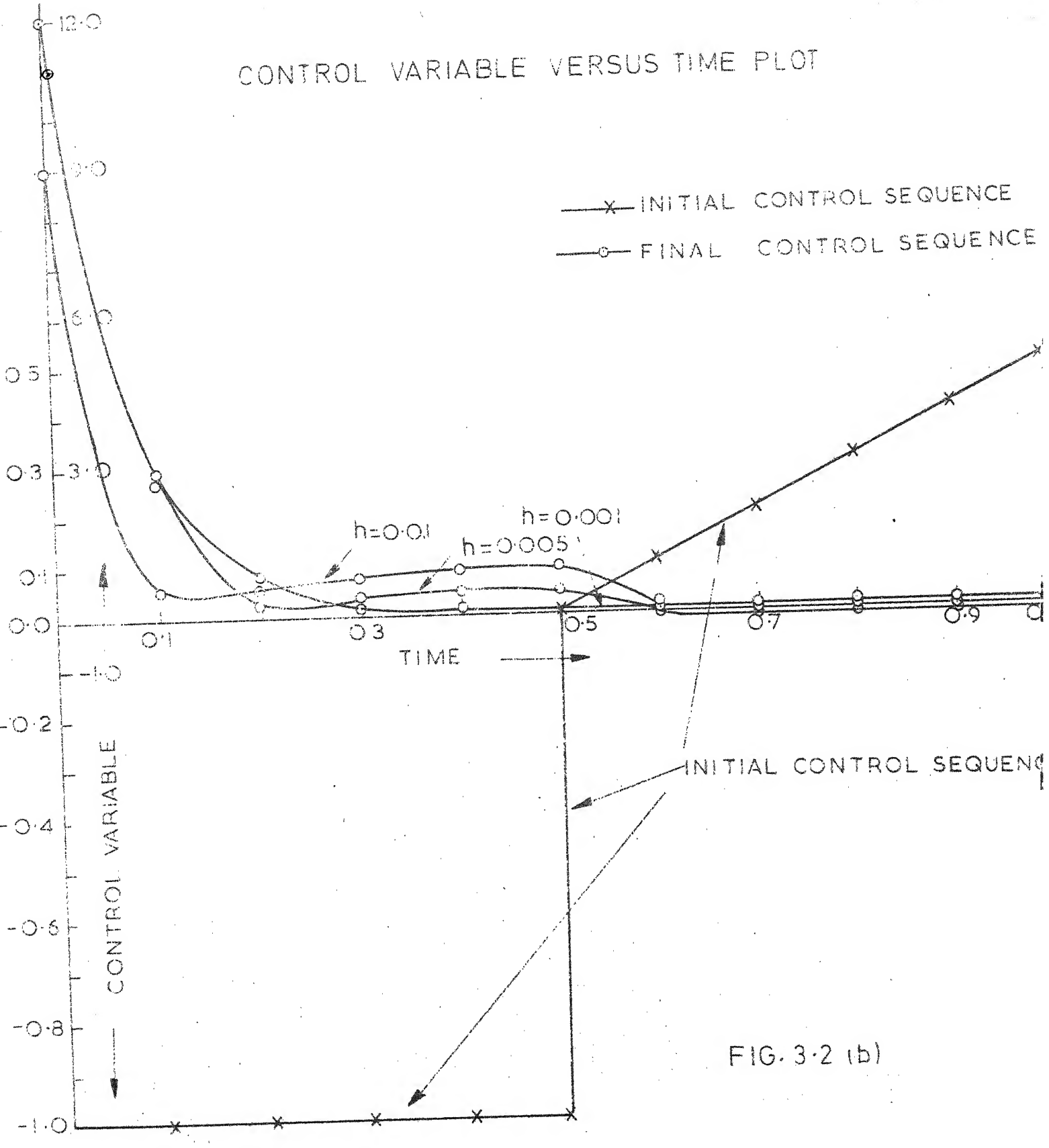

CONTROL VARIABLE VERSUS TIME PLOT
INITIAL CONTROL SEQUENCE
FINAL CONTROL SEQUENCE
h=0·01
h=0·005
h=0·001
TIME
CONTROL VARIABLE
INITIAL CONTROL SEQUENCE
12·0
9·0
6·0
3·0
0·5
0·3
0·1
0·0
-0·2
-0·4
-0·6
-0·8
-1·0
0·1
0·3
0·5
0·7
0·9

FIG. 3·2 (b)

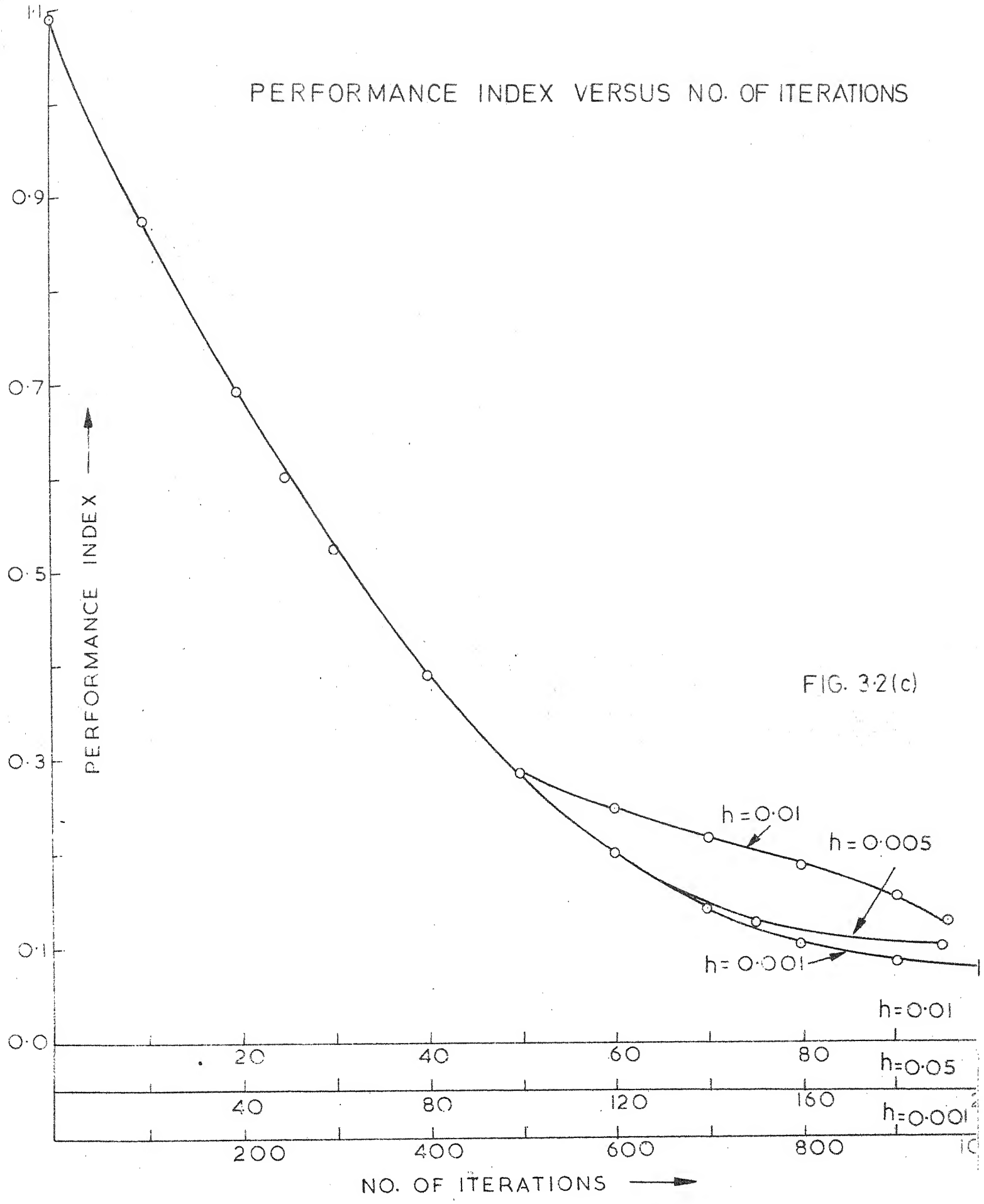

PERFORMANCE INDEX VERSUS NO. OF ITERATIONS
1·1
0·9
0·7
0·5
0·3
0·1
0·0
PERFORMANCE INDEX
h = 0·01
h = 0·005
h = 0·001
h=0·01
h=0·05
h=0·001
20
40
60
80
40
80
120
160
200
400
600
800
NO. OF ITERATIONS

FIG. 3·2(c)

for doing this was to see whether a faster convergence to a solution could be achieved. In the original MLV algorithm, if at any iteration, the kth point of current trajectory was perturbed by +h say, and this led to a 'success', then the state value, control sequence and the incremental performance index values corresponding to the perturbation, replaced the old state value, control sequence and incremental performance index values respectively. Then the algorithm went to the (k+1)th point and repeated the local variations procedure till the whole trajectory was swept. The case where a perturbation of -h was a 'success' was similarly dealt with. In the modified version of the algorithm, if, at the kth point of the current trajectory a perturbation of +h is a 'success' then, instead of replacing the old state value, control sequence and incremental performance index values by the values corresponding to the perturbation and going to the (k+1)th point, a perturbation of +2h is now given to the kth point. If this is a 'success' also, a perturbation of +3h is given to the same point and so on till a maximum perturbation of +5h. Thus, within a limit of +5h, the state, control and incremental performance index values corresponding to the 'most successful' perturbation replace the old values. Then only does the algorithm go to the (k+1)th point. This procedure is

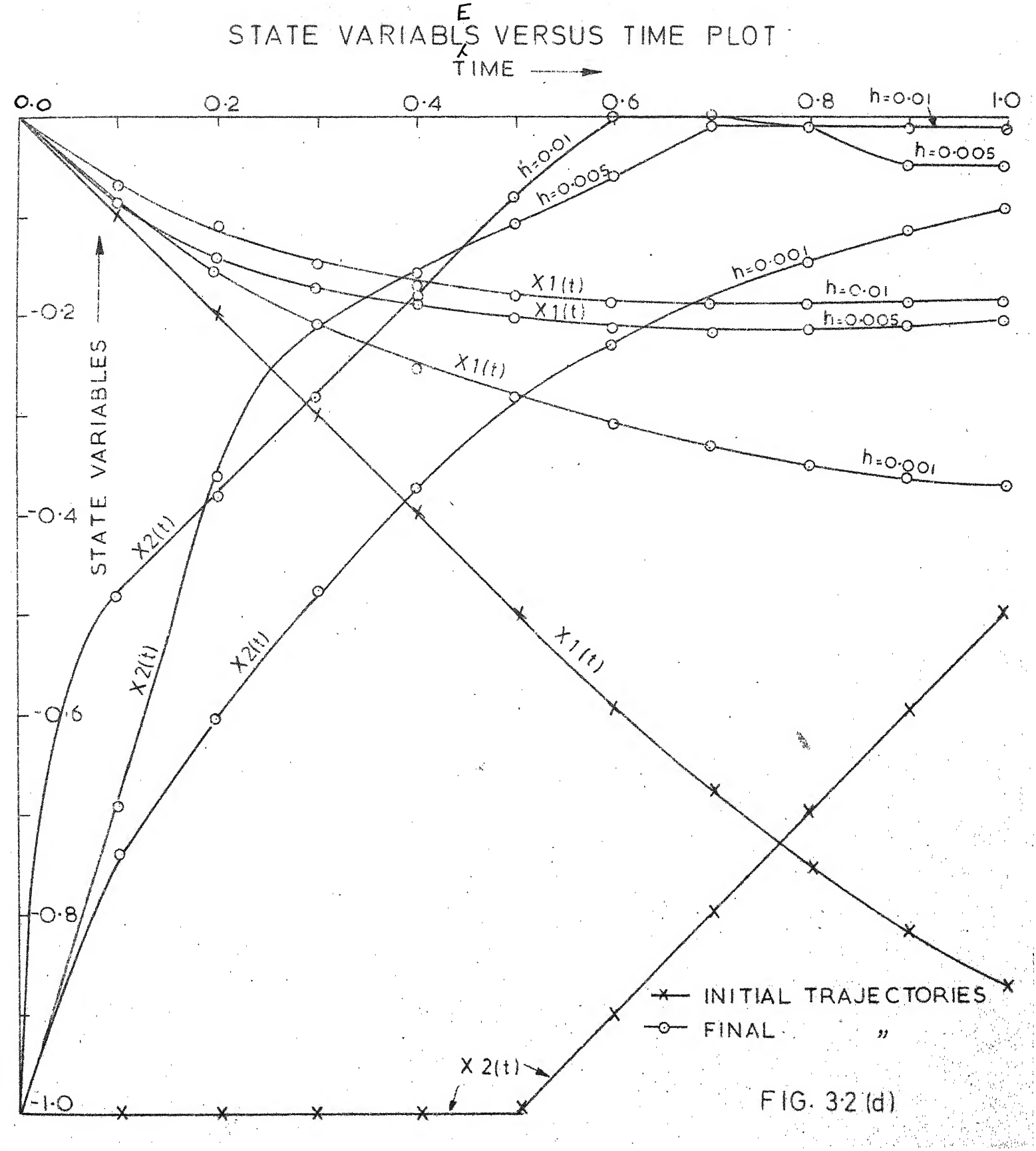

STATE VARIABLES VERSUS TIME PLOT
TIME
STATE VARIABLES
0·0
0·2
0·4
0·6
0·8
1·0
-0·2
-0·4
-0·6
-0·8
-1·0
h=0·01
h=0·005
h=0·001
X1(t)
X2(t)
INITIAL TRAJECTORIES
FINAL "

FIG. 3·2 (d)

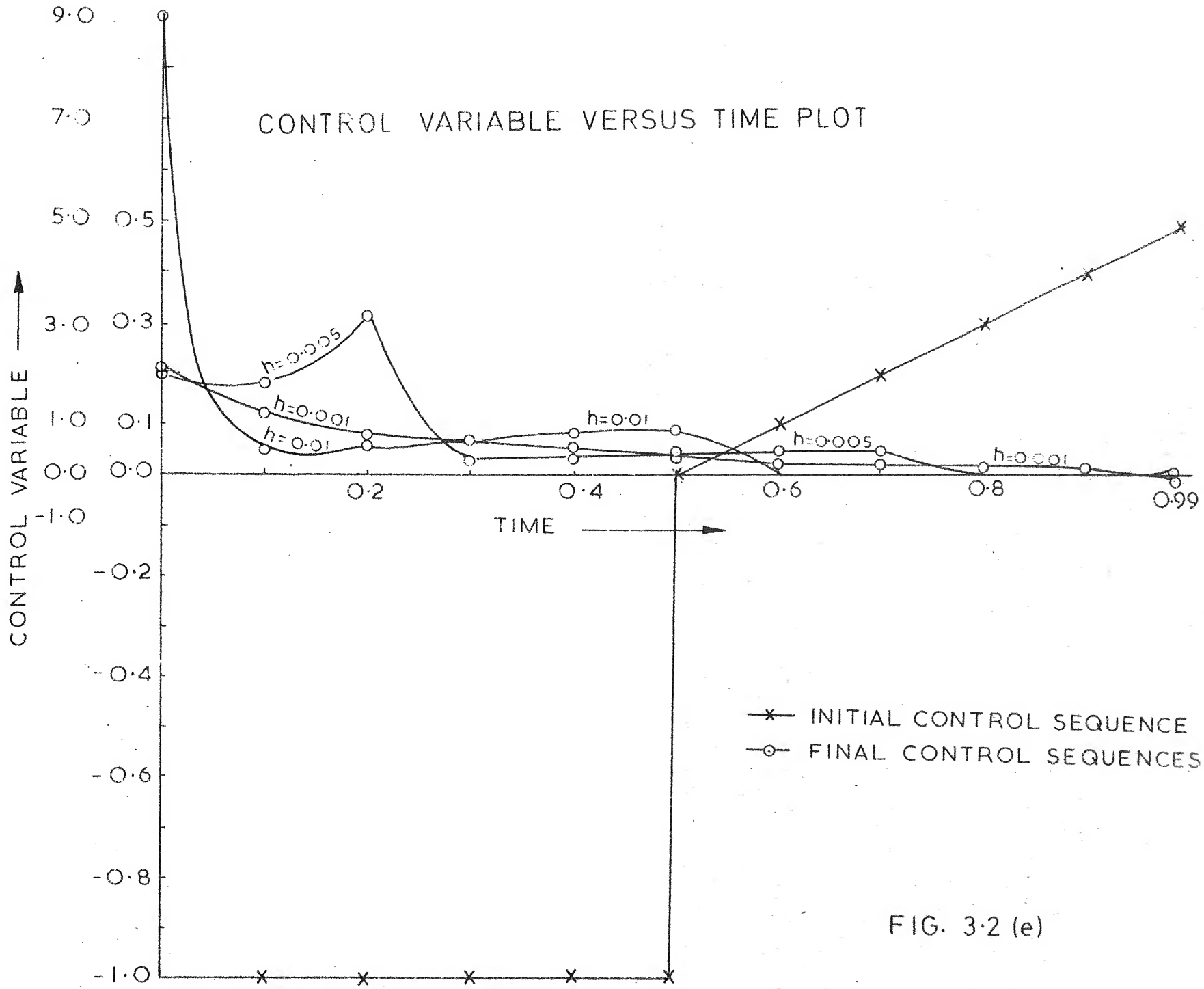

CONTROL VARIABLE VERSUS TIME PLOT
CONTROL VARIABLE
TIME
9·0
7·0
5·0
3·0
1·0
0·0
-1·0
0·5
0·3
0·1
0·0
-0·2
-0·4
-0·6
-0·8
-1·0
0·2
0·4
0·6
0·8
0·99
h=0·005
h=0·001
h=0·01
h=0·01
h=0·005
h=0·001
INITIAL CONTROL SEQUENCE
FINAL CONTROL SEQUENCES

FIG. 3·2 (e)

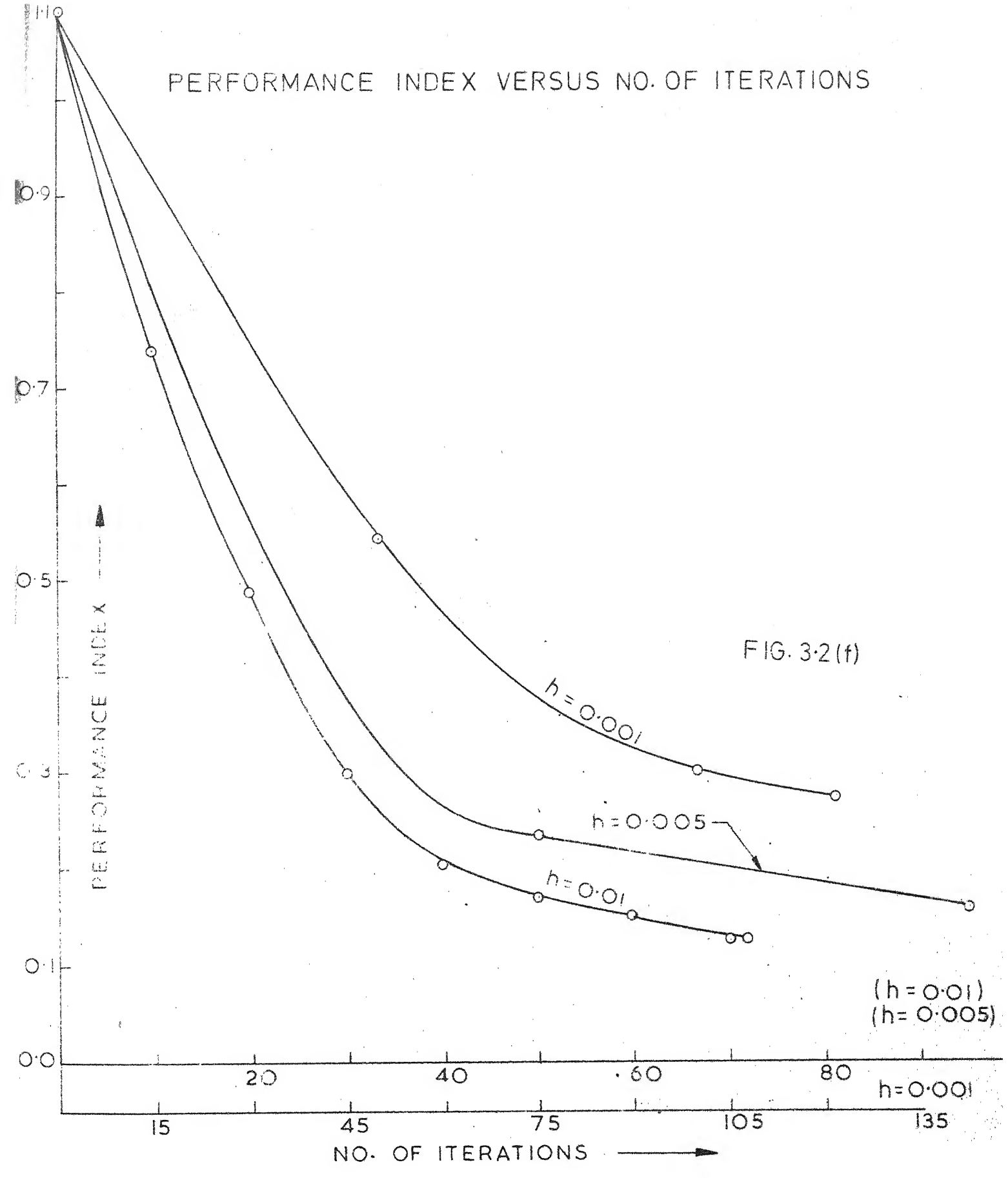

PERFORMANCE INDEX VERSUS NO. OF ITERATIONS
1·0
0·9
0·7
0·5
0·1
0·0
PERFORMANCE INDEX
h = 0·001
h = 0·005
h = 0·01
(h = 0·01)
(h = 0·005)
h = 0·001
20
40
60
80
15
45
75
105
135
NO. OF ITERATIONS

FIG. 3·2 (f)

the modification. Another fact to be noted is that whereas h = 0.001 produced the smallest value of J_{min} and h = 0.01 the largest when the original algorithm was used, vice versa is true when the modified algorithm was used, at least in the present case. The best result achieved by the modified algorithm using h = 0.01 is, however, not as good as that obtained by Lasdon et al.[25].

3.3 Linear, Two-State Variable, Constrained, Optimal Control Problems

Problem 3

The first problem of the above category chosen was:

State Equations: $$\dot{x}_1 = x_2 \qquad (3.3\text{-}1)$$

$$\dot{x}_2 = -x_2 + u \qquad (3.3\text{-}2)$$

It is required to minimise the performance index

$$J = \int_0^1 [x_1^2 + x_2^2 + 0.005\, u^2]\, dt \qquad (3.3\text{-}3)$$

subject to

State Constraint: $x_2(t) - 8(t - 0.5)^2 + 0.5 \leqslant 0$

and the

Initial Conditions: $x_1(o) = 0$

$$x_2(o) = -1$$

This problem was originally considered by Jacobson and Lele [42] using the Generalised Gradient Method wherein, the gradient of J is calculated with respect to a set of independent variables rather than with respect to control. In this method, the choice of independent variables is dictated by the constraints on the problem and could result in different combinations of state and control variables as independent variables along different parts of the trajectory. Here, the directions of search are determined using gradient projection and the conjugate gradient methods. Mehra and Davis [43] have used some of the state variables, instead of the control variable, as the independent variables to solve the above problem. They have also indicated the possibility of choosing a mixed set of independent variables consisting of state and control variables. The switchover from one set of independent variables to another occurs when the constraint boundary is hit.

3.3.1 Solution with the original MLV Algorithm

The constraint on $x_2(t)$ is a state-variable inequality constraint. In the application of MLV to this problem, the state variable $x_2(t)$ is given perturbations. The perturbed values of $x_1(t)$ and $u(t)$ would solely depend on the perturbations given to $x_2(t)$. An initial feasible $x_2(t)$ trajectory was chosen

as the starting point of the solution such that the corresponding $x_1(t)$ trajectory and the control history $u(t)$, in $[0,1]$, are admissible. Discretisation was done as in Section 3.2.1, with N = 100 and $\Delta t = 0.01$. Then MLV was applied to obtain a locally optimal solution using the same stopping criterion as in Section 3.2.1. Due to the presence of a state constraint, the choice of an initial feasible $x_2(t)$ trajectory becomes important. The initial feasible state trajectories of Section 3.2.1 happened to satisfy all the constraints and were, thus, chosen for this problem also. The results obtained by MLV and Mehra and Davis [43] have been tabulated in Table 3.3(a) while the state trajectories, control sequences, and performance index versus iteration number plots have been shown in Figures 3.3(a), (b) and (c).

Initial value of performance index = J_{ini} = 1.09368600

Sl. No.	Perturbation step h	No. of Iterations K	Min. value of performance index J_{min}	Execution time on IBM 7044 (secs)	Optimum value of performance index, (Mehra and Davis[43])
1	0.01	56	0.30021131	10.00	
2	0.005	161	0.23267423	19.00	$\approx$ 0.17
3	0.001	898	0.21874910	94.00	

Table 3.3(a)

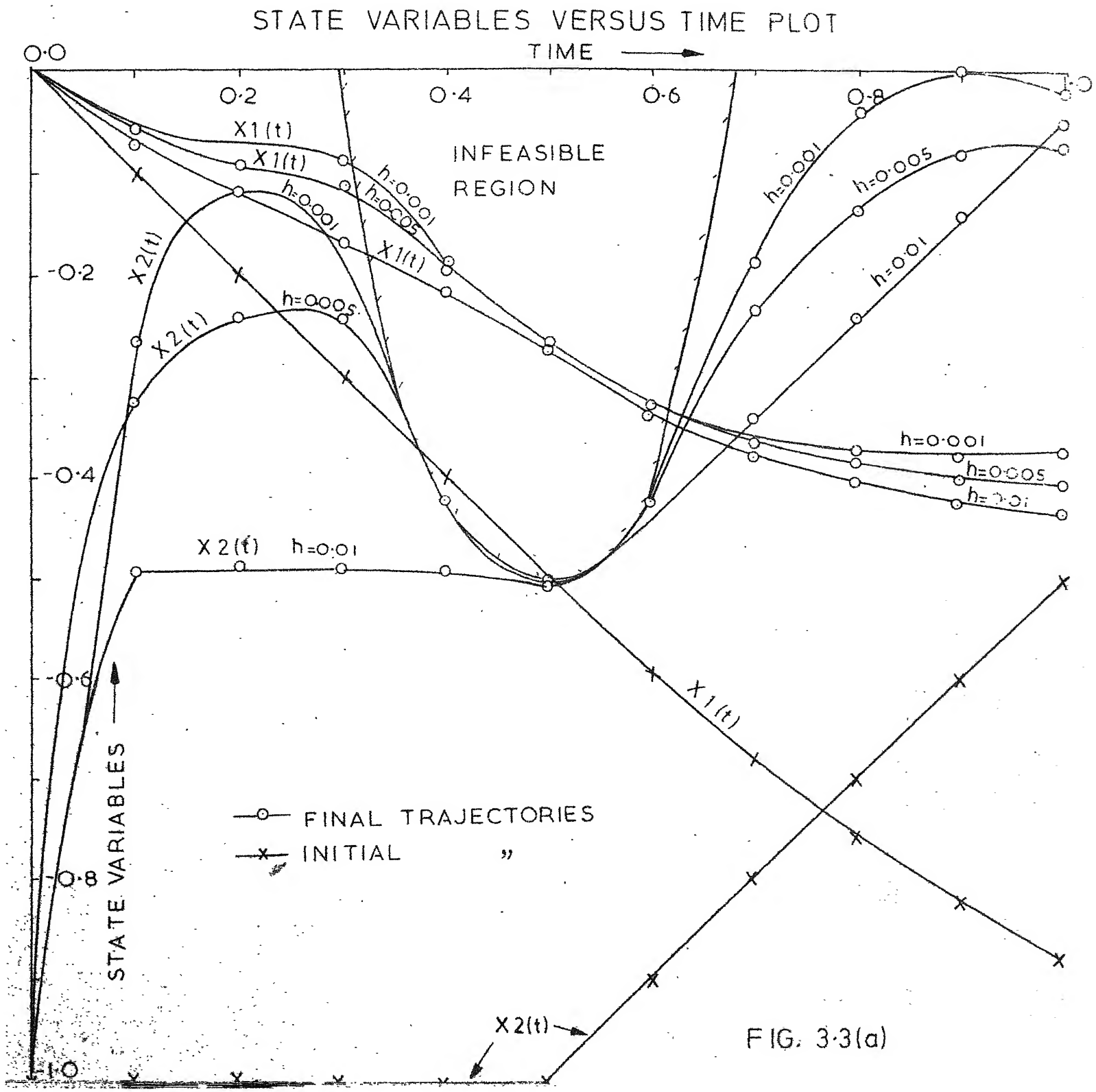

STATE VARIABLES VERSUS TIME PLOT
TIME
0·0
0·2
0·4
0·6
0·8
1·0
-0·2
-0·4
-0·6
-0·8
-1·0
STATE VARIABLES
INFEASIBLE REGION
X1(t)
X2(t)
h=0·001
h=0·005
h=0·01
X2(t) h=0·01
FINAL TRAJECTORIES
INITIAL "

FIG. 3·3(a)

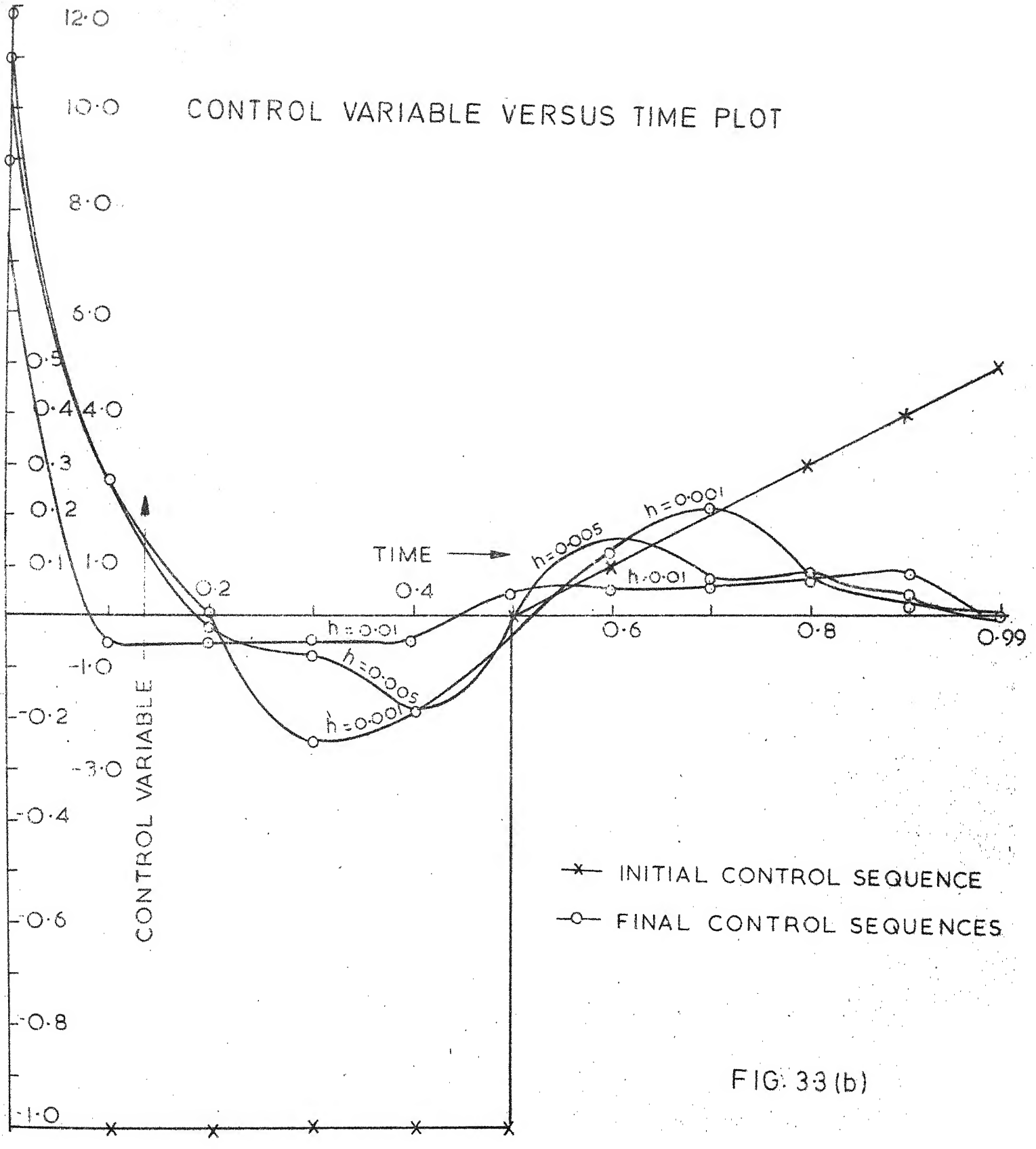

CONTROL VARIABLE VERSUS TIME PLOT
12·0
10·0
8·0
6·0
4·0
3·0
2·0
1·0
0·9
0·4
0·3
0·2
0·1
-1·0
-0·2
-0·4
-0·6
-0·8
-1·0
CONTROL VARIABLE
TIME
0·2
0·4
0·6
0·8
0·99
h=0·01
h=0·005
h=0·001
h=0·005
h=0·001
h=0·01
INITIAL CONTROL SEQUENCE
FINAL CONTROL SEQUENCES

FIG. 3·3 (b)

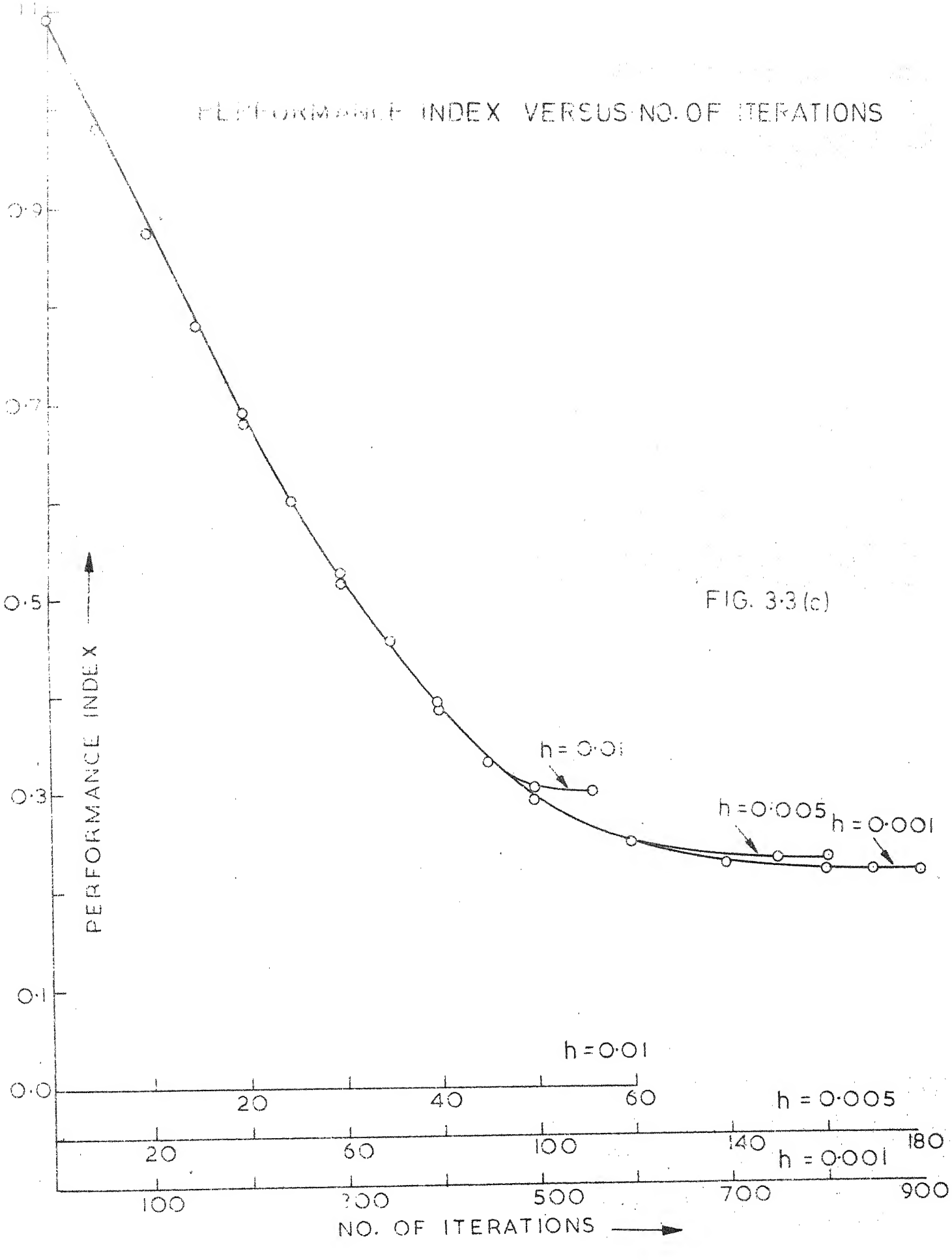

PERFORMANCE INDEX VERSUS NO. OF ITERATIONS
PERFORMANCE INDEX
0·9
0·7
0·5
0·3
0·1
0·0
h = 0·01
h = 0·005
h = 0·001
20
40
60
20
60
100
140
180
100
300
500
700
900
NO. OF ITERATIONS

FIG. 3·3 (c)

Jacobson and Lele [42] used a slack variable to transform the constrained problem into an unconstrained one. The MLV technique does not require such a transformation and considers the inequality constraint as it is. This reduces the complexity of solution.

Although Mehra and Davis [43] obtained a lower value of J_{min} than the results achieved by MLV, their $x_2(t)$ trajectory did not satisfy the initial condition. Using MLV, the best results were achieved with h = 0.001 in 898 iterations, and all the constraints were satisfied. In order to ease comparison with the results obtained by Mehra and Davis [43], Figures 3.3 (g), (h) and (i), giving the state trajectory, control sequence, and the performance index versus iteration number plots respectively, for h = 0.001, have been drawn.

3.3.2 Solution with a modification of the original MLV Algorithm

Problem 3 has been solved with a slightly modified version of the original MLV technique. The modification consists of the following:

a) An initial perturbation step h was chosen.

b) This h was utilised to find the final state trajectory and control sequence, the performance index value and the number of iterations required, with the same initial feasible state trajectory and control sequence as in Section 3.3.1.

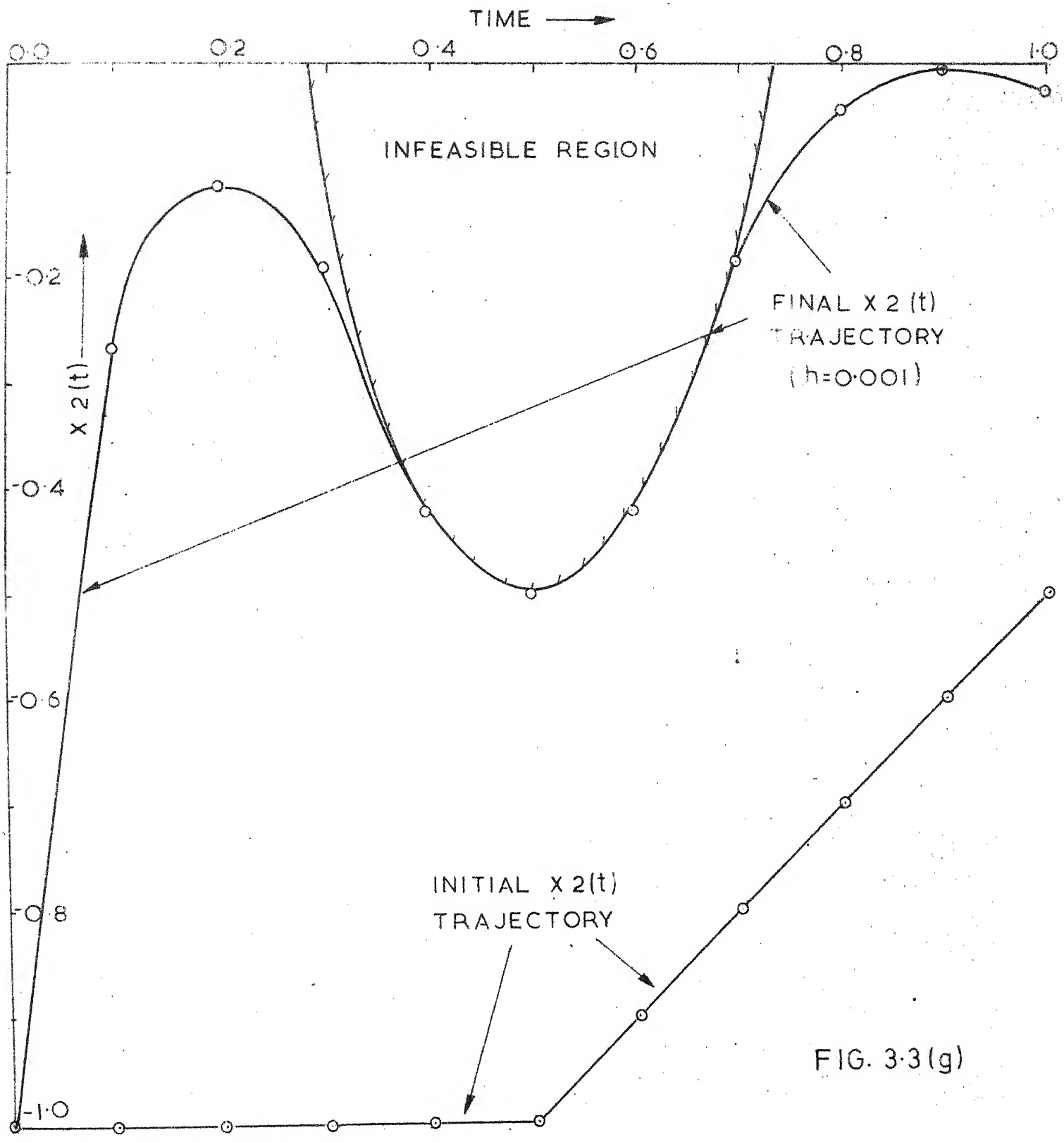

X 2 (t) VERSUS TIME PLOT
TIME
0·0
0·2
0·4
0·6
0·8
1·0
INFEASIBLE REGION
FINAL X 2 (t)
TRAJECTORY
(h=0·001)
X 2 (t)
-0·2
-0·4
-0·6
-0·8
-1·0
INITIAL X 2(t)
TRAJECTORY

FIG. 3·3 (g)

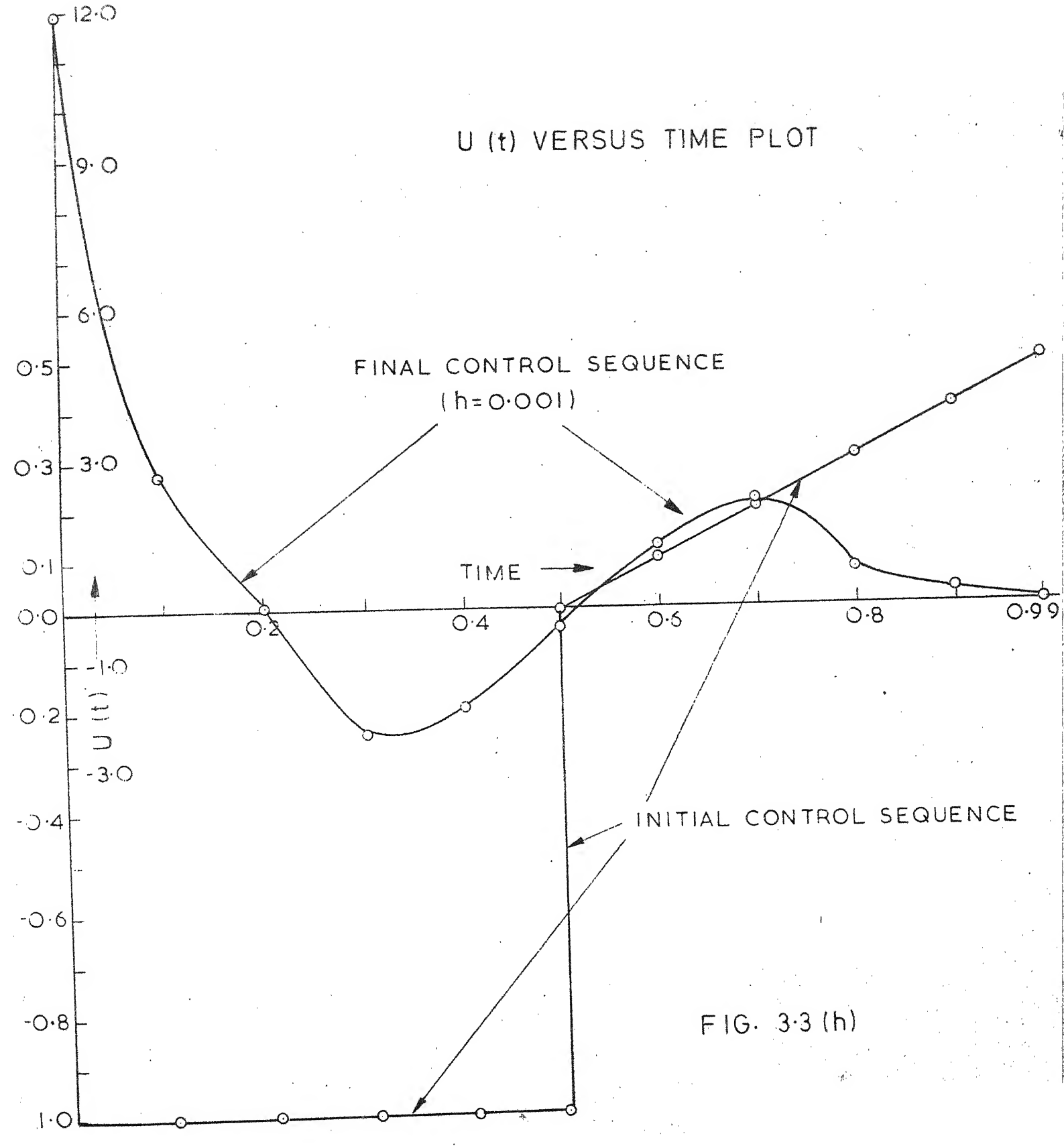

U (t) VERSUS TIME PLOT
12·0
9·0
6·0
3·0
0·5
0·3
0·1
0·0
-1·0
-3·0
0·2
0·4
0·6
0·8
0·99
TIME
U (t)
FINAL CONTROL SEQUENCE
(h=0·001)
INITIAL CONTROL SEQUENCE

FIG. 3·3 (h)

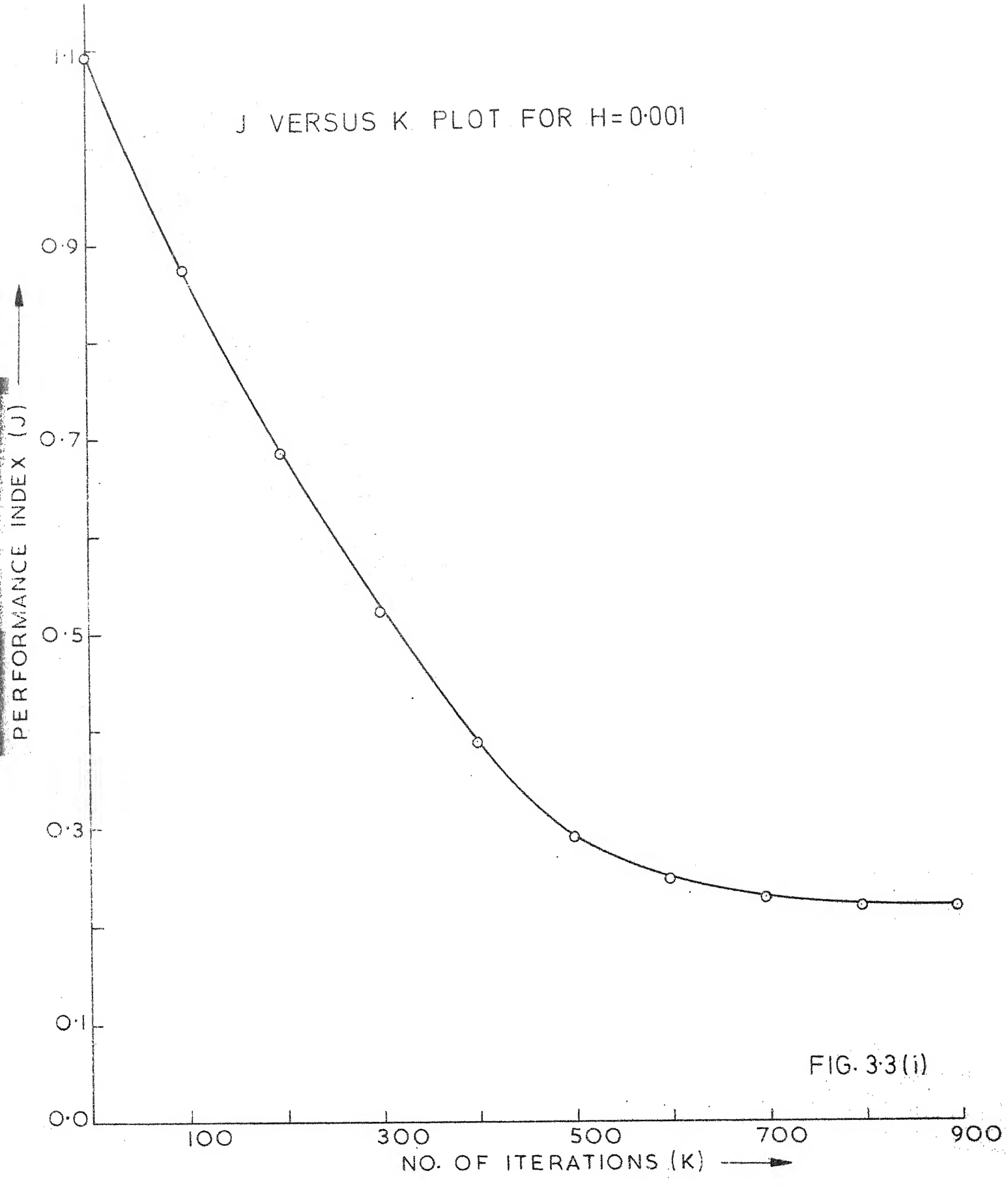

J VERSUS K PLOT FOR H=0·001
PERFORMANCE INDEX (J)
1·1
0·9
0·7
0·5
0·3
0·1
0·0
100
300
500
700
900
NO. OF ITERATIONS (K)

FIG. 3·3(i)

c) h was then halved and the final state trajectory and control sequence of b) above, became the initial feasible state trajectory and control sequence. The MLV algorithm was then used with this h and these initial values to generate the corresponding final state trajectory and control sequence.

The procedure c) above was repeated till a stopping criterion, depending on the h-value used in Table 3.3(a) of Section 3.3.1 with which we are comparing this modified method, was satisfied. Then the final state trajectories, control sequence, performance index value and the total number of iterations required were printed out.

Double precision arithmetic was used in the computations and the results for the above modification to MLV and the original MLV, with h = 0.005 and h = 0.001, are presented below for comparison.

Initial value of performance index = J_{ini} = 1.093687 (double precision result)

Original MLV Algorithm (with double precision arithmetic)

Sl. No.	Perturbation step h	No. of Iterations K	Min. value of performance index J_{min}	Execution time on IBM 7044 (secs)
1	0.005	161	0.2321809	42.00
2	0.001	898	0.2187232	216.0

Table 3.3(b)

Comparing with Table 3.3(a), for h = 0.005 and h=0.001, it is seen that although the number of iterations required for convergence remains exactly the same and the value of J_{min} almost the same, the execution times have increased considerably due to double precision arithmetic. A program listing, giving a solution to Problem 3 with double precision arithmetic and h = 0.001, has been provided.

MLV Algorithm modified as described above.

Sl. No.	Initial perturbation step h_i	Final perturbation step h_f	Total No. of iterations K	Min.value of performance index J_{min}	Execution time on IBM 7044 (secs.)
1	0.01	0.005	114	0.2321933	33.00
2	0.016	0.001	213	0.1817503	65.00

Table 3.3(c)

Comparing with Tables 3.3(a) and (b), for h = 0.005 and h = 0.001, it is observed that the modification has not only significantly reduced the number of iterations required for convergence as well as the execution time, but it has also reduced the value of J_{min} for h = 0.001. The value of J_{min} for h = 0.005, however, remains almost the same. There could be some other way of choosing the step size from iteration to iteration which could further reduce J_{min} as well as computation time.

```
C       A SOLUTION TO PROBLEM3 WITH DOUBLE PRECISION ARITHMETIC AND H=0.001
C       THE INITIAL NOMINAL TRAJECTORIES AND  CONTROL SEQUENCE ARE FED IN
C
        DOUBLE PRECISION X1(101),X2(101),U(100),DELCST(100)
        DOUBLE PRECISION PRINDX,A,B
        X1(1)=0.
        X2(1)=-1.
        DO 1 I=1,50
        X2(I+1)=X2(I)
        X1(I+1)=0.01*X2(I)+X1(I)
      1 CONTINUE
        X1(52)=0.01*X2(51)+X1(51)
        X2(52)=-0.99
        DO 2 I=52,100
        X2(I+1)=X2(I)+0.02
        X1(I+1)=0.01*X2(I)+X1(I)
      2 CONTINUE
        DO 6 I=1,100
        U(I)=100.*X2(I+1)-99.*X2(I)
      6 CONTINUE
C
        PRINT 100
  100   FORMAT(15X,*THE INITIAL NOMINAL TRAJECTORIES ARE*)
        PRINT 101,X1
  101   FORMAT(2X,10D13.5)
        PRINT 101,X2
        PRINT 200
  200   FORMAT(15X,*THE INITIAL NOMINAL CONTROL SEQUENCE IS*)
        PRINT 101,U
C
C       CALCULATION OF PERFORMANCE INDEX
C
        PRINDX=0.0
        DO 3 I=1,100
        DELCST(I)=0.01*(X1(I)*X1(I)+X2(I)*X2(I))+0.00005*U(I)*U(I)
        PRINDX=PRINDX+DELCST(I)
      3 CONTINUE
        PRINT 102,PRINDX
  102   FORMAT(10X,*THE INITIAL VALUE OF P.I. =*,D15.7)
C
        K=1
        A=PRINDX
     10 CALL ELEOPR(X1,X2,U,DELCST)
        B=A
        A=0.0
        DO 4 I=1,100
        A=A+DELCST(I)
      4 CONTINUE
        IF(K-K/50*50)302,301,302
  301   PRINT 103,K,A
```

```
  103 FORMAT(10X,*ITERATION NO. =*,I5,14X,*COST =*,D15.7)
  302 B=(B-A)/B
      IF(B.LT.1.D-6)GOTO5
      K=K+1
      GOTO10
C
    5 PRINT 104
  104 FORMAT(15X,*THE FINAL STATE TRAJECTORIES ARE GIVEN BY*)
      PRINT 101,X1
      PRINT 101,X2
      PRINT 105
  105 FORMAT(15X,*THE FINAL CONTROL SEQUENCE IS GIVEN BY*)
      PRINT 101,U
      PRINT 106,K,A
  106 FORMAT(15X,*THE FINAL NO. OF ITERATIONS =*,I5//14X,
     1 *THE FINAL VALUE OF P.I. =*,D15.7)
      STOP
      END
      SUBROUTINE ELEOPR(X1,X2,U,DELCST)
C     THIS S/R PERFORMS THE ELEMENTARY OPERATION
C
      DOUBLE PRECISION X1(101),X2(101),U(100),DELCST(100)
      DOUBLE PRECISION H,SAVE,X1CHK,X2CHK,UMINUS,UPLUS
      DOUBLE PRECISION DLCST1,DLCST2,A,B,C,Z
      DO 1 I=2,101
      H=0.001
      SAVE=X2(I)
      J=1
C
    2 X2CHK=SAVE+H
      Z=I-1
      C=8.*(Z*0.01-0.5)*(Z*0.01-0.5)-0.5
      IF((X2CHK-C).GT.0.)GOTO4
      UMINUS=100.*X2CHK-99.*X2(I-1)
      DLCST1=0.01*(X1(I-1)*X1(I-1)+X2(I-1)*X2(I-1))
     1 +0.00005*UMINUS*UMINUS
      IF(I.EQ.101)GOTO5
C
      UPLUS=100.*X2(I+1)-99.*X2CHK
      DLCST2=0.01*(X1(I)*X1(I)+X2CHK*X2CHK)
     1 +0.00005*UPLUS*UPLUS
C
      A=DLCST1+DLCST2
      B=DELCST(I-1)+DELCST(I)
      IF(A.GT.B)GOTO4
      X1(I)=0.01*X2(I-1)+X1(I-1)
      X2(I)=X2CHK
      N=I-1
      U(N)=UMINUS
      U(I)=UPLUS
```

```
      DELCST(N)=DLCST1
      DELCST(I)=DLCST2
      GOTO1
C
    4 J=J+1
      IF(J.GT.2)GOTO1
      H=-H
      GOTO2
C
    5 A=DLCST1
      B=DELCST(I-1)
      IF(A.GT.B)GOTO4
      X1(I)=0.01*X2(I-1)+X1(I-1)
      X2(I)=X2CHK
      N=I-1
      U(N)=UMINUS
      DELCST(N)=A
    1 CONTINUE
      RETURN
      END
```

3.3.3 Solution with the same modification as in Section 3.2.2.

Finally, Problem 3 was solved with the modification detailed in Section 3.3.2. Figures 3.3(d), (e) and (f) show the state trajectories, control sequences, and the performance index versus iteration number plots respectively. From the results tabulated in Table 3.3(d), similar conclusions as in Section 3.2.2 can be drawn.

Initial value of performance index = J_{ini} = 1.094

Sl. No.	Perturbation step h	No.of iterations K	Min.value of performance index J_{min}	Execution time on IBM 7044 (secs)
1	0.01	56	0.3054	14.00
2	0.005	88	0.3125	22.00
3	0.001	102	0.3470	46.00

Table 3.3(d)

Problem 4

The second problem in the category being considered, and the last problem chosen in this chapter was:

State Equations:
$$\dot{x}_1 = x_2 \qquad (3.3\text{-}4)$$
$$\dot{x}_2 = -x_2 + u \qquad (3.3\text{-}5)$$

It is required to minimise the performance index

$$J = \int_0^1 [x_1^2 + x_2^2 + 0.005\, u^2]dt \qquad (3.3\text{-}6)$$

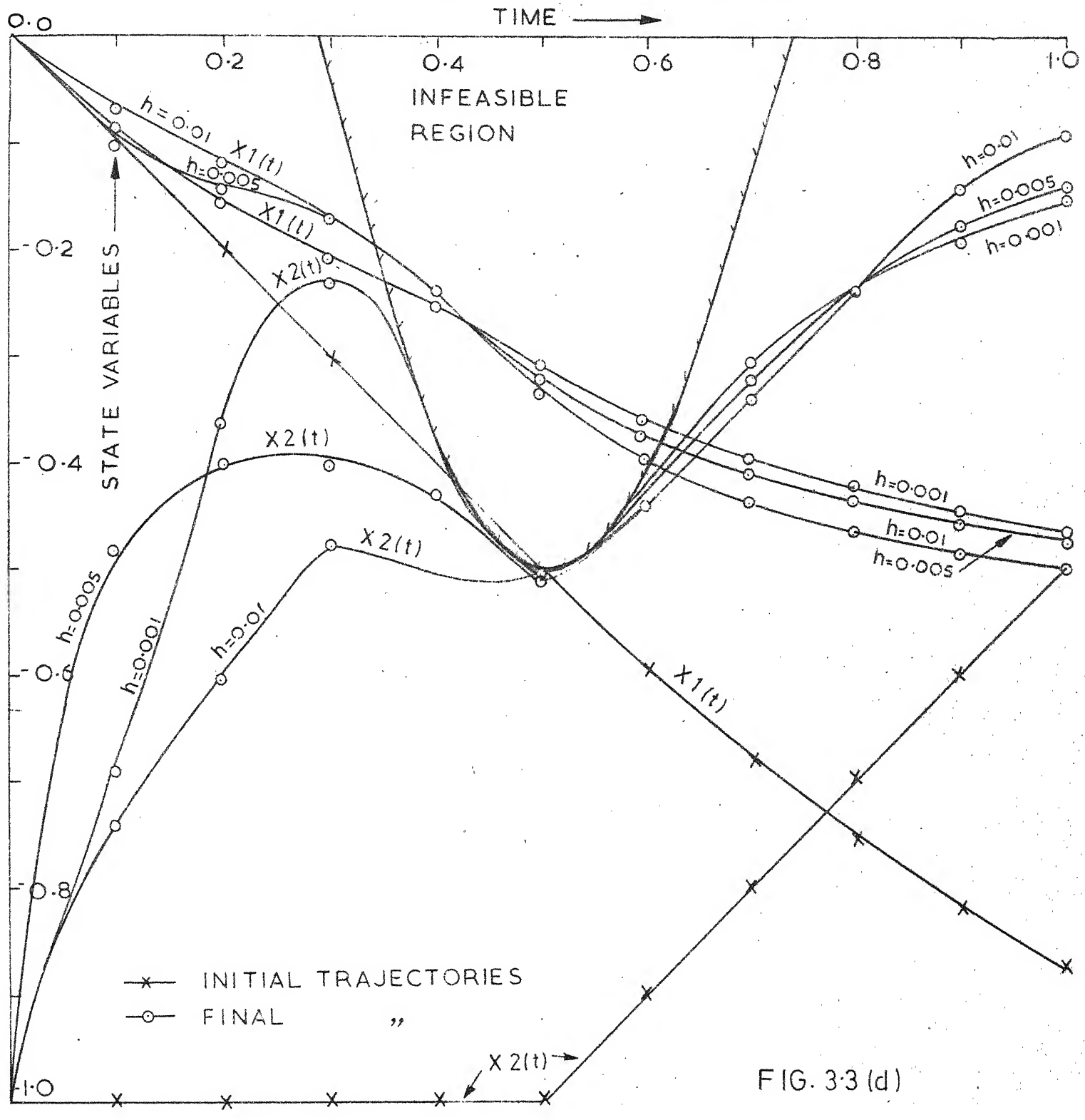

STATE VARIABLES VERSUS TIME PLOT
TIME
0·0
0·2
0·4
0·6
0·8
1·0
INFEASIBLE
REGION
STATE VARIABLES
h=0·01
X1(t)
h=0·005
X1(t)
X2(t)
X2(t)
X2(t)
h=0·005
h=0·001
h=0·01
h=0·01
h=0·005
h=0·001
h=0·001
h=0·005
h=0·01
X1(t)
X2(t)
INITIAL TRAJECTORIES
FINAL ,,

FIG. 3·3 (d)

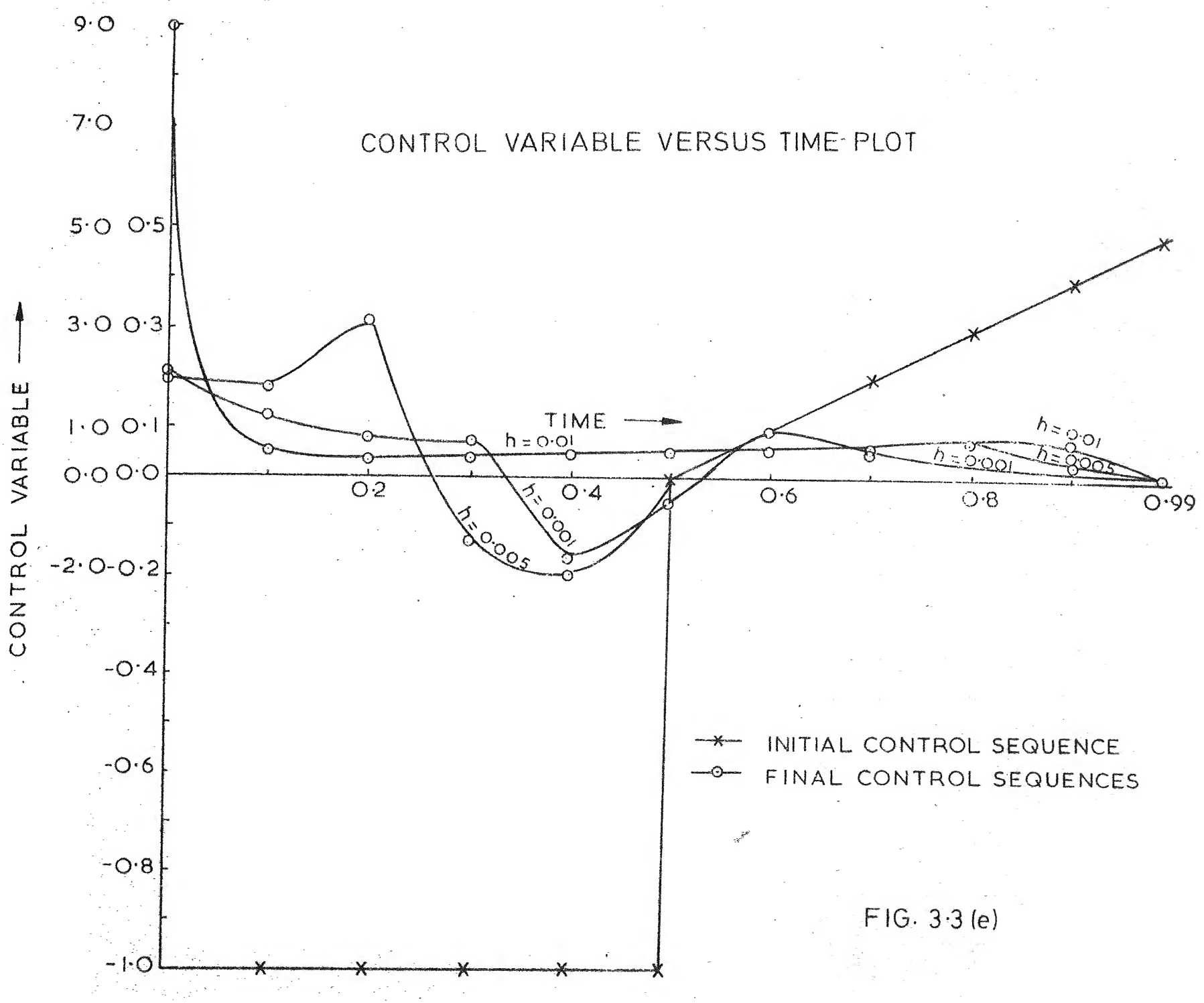

CONTROL VARIABLE VERSUS TIME-PLOT
CONTROL VARIABLE
TIME
9·0
7·0
5·0
0·5
3·0
0·3
1·0
0·1
0·0
0·0
-2·0
-0·2
-0·4
-0·6
-0·8
-1·0
0·2
0·4
0·6
0·8
0·99
h = 0·01
h = 0·001
h = 0·005
h = 0·01
h = 0·001
h = 0·005
INITIAL CONTROL SEQUENCE
FINAL CONTROL SEQUENCES

FIG. 3·3 (e)

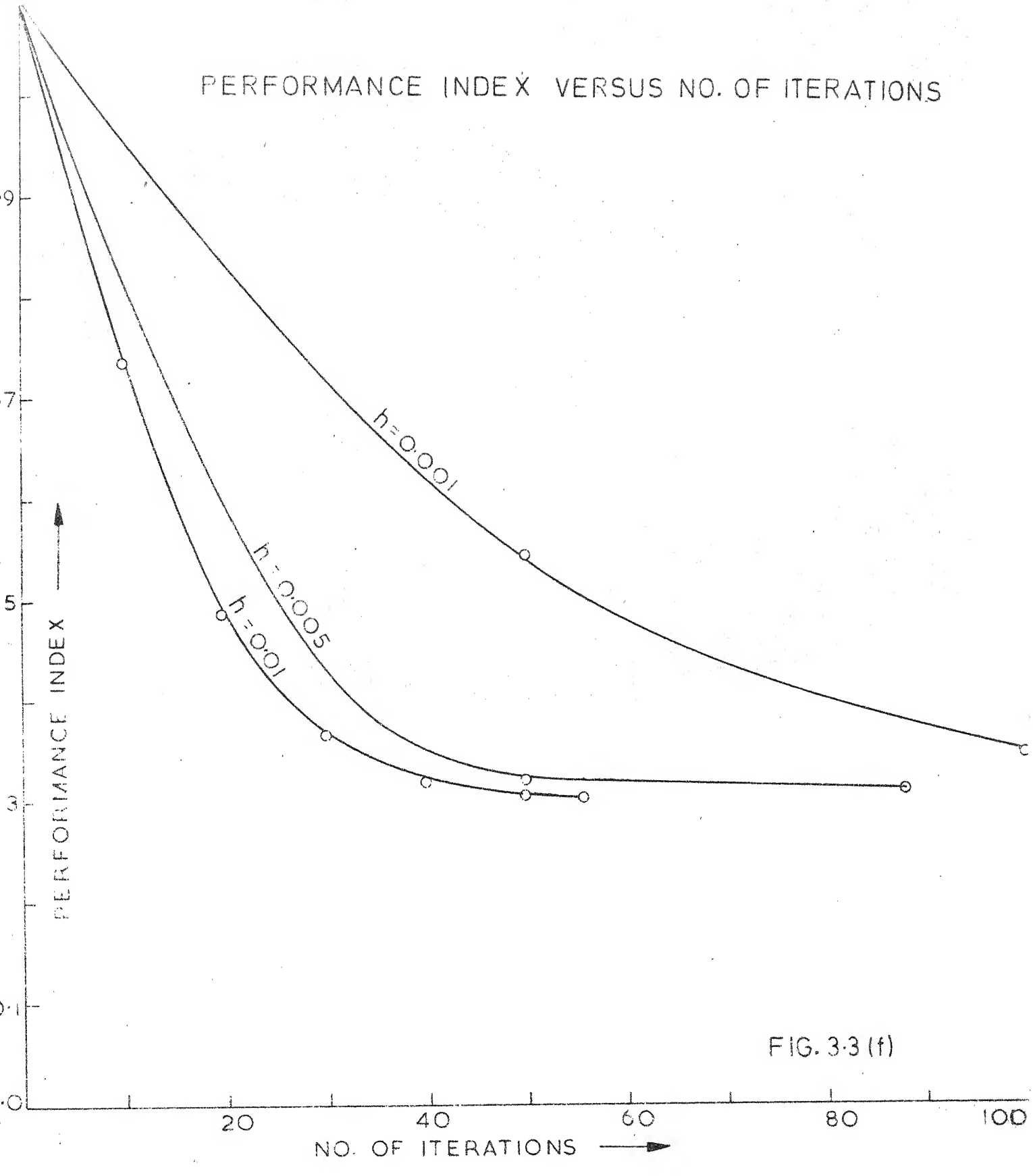

PERFORMANCE INDEX VERSUS NO. OF ITERATIONS
h=0.001
h=0.005
h=0.01
PERFORMANCE INDEX
NO. OF ITERATIONS
20
40
60
80
100

FIG. 3·3 (f)

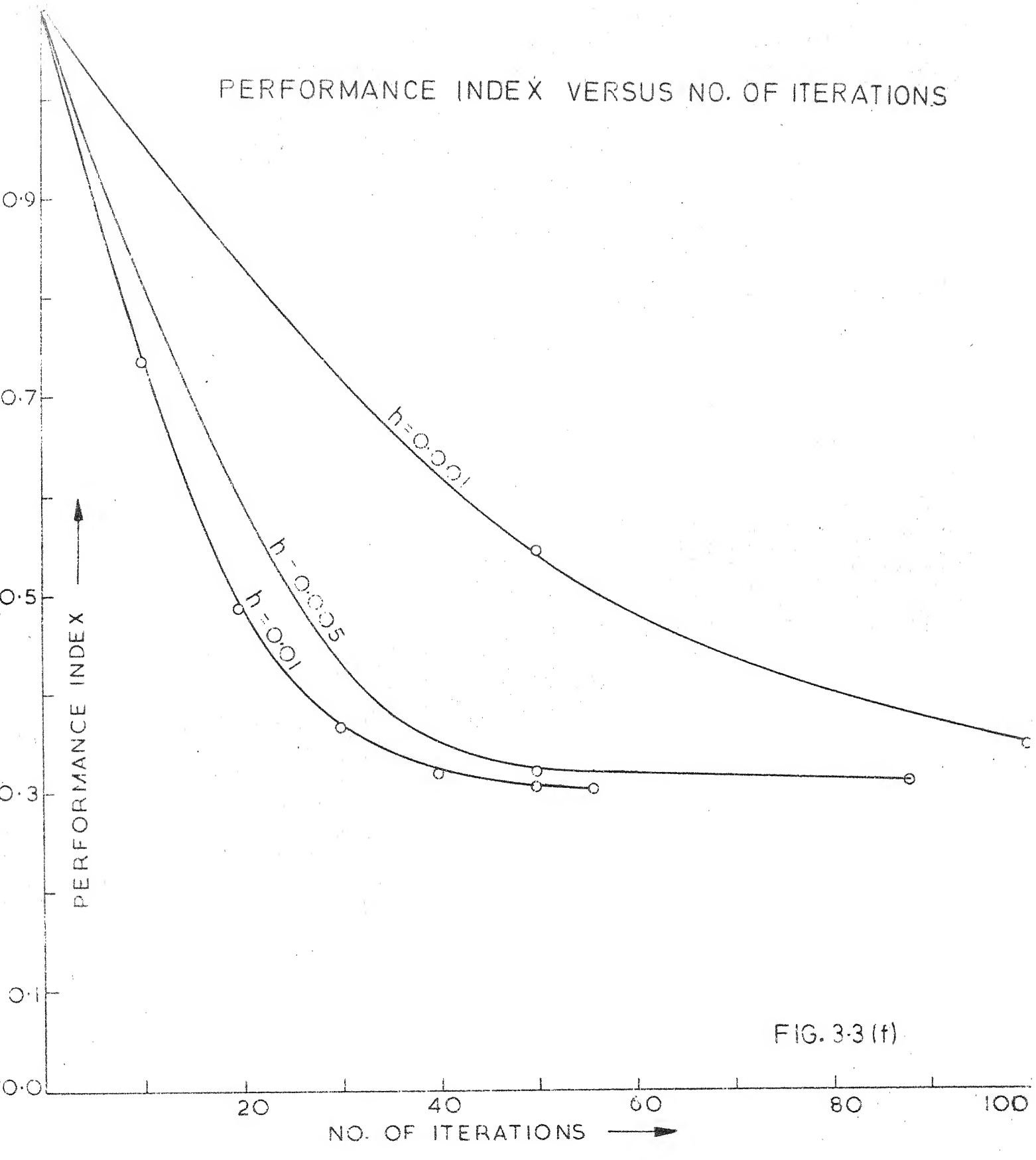

PERFORMANCE INDEX VERSUS NO. OF ITERATIONS
h=0.001
h=0.005
h=0.01
PERFORMANCE INDEX
0·9
0·7
0·5
0·3
0·1
0·0
20
40
60
80
100
NO. OF ITERATIONS

FIG. 3-3 (f)

subject to

State Constraint: $x_1(t) - 8(t - 0.5)^2 + 0.5 \leq 0$

and the

Initial Conditions: $x_1(o) = 0$

$x_2(o) = -1$

This problem too, was originally considered by Jacobson and Lele [42] and has been solved by Mehra and Davis [43] using the Generalised Gradient Method.

3.3.4 Solution with the original MLV Algorithm

The problem being considered is the same as Problem 3 in Section 3.3 with the exception that the state inequality constraint is now on $x_1(t)$ instead of $x_2(t)$. For application of the MLV algorithm the problem was put into the discrete form as earlier, and a particular control sequence $u(k)$; $(k = 0,1,\ldots,N-1)$, was chosen such that the corresponding state trajectories were feasible. These intial feasible state trajectories were different from those in Problem 3, as shown in Fig. 3.3(j).

Discretisation was done as in Section 3.2.1 with $N = 100$ and $\Delta t = 0.01$. The MLV algorithm was applied, with the same stopping criterion as in Section 3.2.1, to obtain a locally optimal solution. Double precision arithmetic was used for greater accuracy and the results

obtained have been tabulated in Table 3.3(e) together with the results obtained by Mehra and Davis [43]. Figures 3.3 (j), (k) and (l) show the nature of the state trajectories, control sequences, and the performance index versus iteration number plots respectively.

Initial value of performance index = J_{ini} = 1.454357

Sl. No.	Perturbation step h	No. of Iterations K	Min.value of performance index J_{min}	Execution time on IBM 7044 (secs)	Optimum value of performance index [43]
1	0.01	72	0.5923687	21.00	
2	0.005	189	0.5146318	48.00	0.792942
3	0.001	1035	0.5245792	254.0	

Table 3.3(e)

Using MLV, the best results were obtained in 189 iterations with h = 0.005. Thus, in this application, the MLV has given a much better result than that obtained by Mehra and Davis [43]. Another aspect to be noted is that Figures 3.3(j) and (k) indicate that the final state trajectories and control sequences obtained with the help of MLV are quite different from those obtained by Mehra and Davis [43], indicating that MLV has picked up a locally optimal solution different from that obtained by Mehra and Davis [43]. In order to ease comparison with the results obtained by Mehra and Davis [43], Figures 3.3 (p), (q) and (r), showing the state trajectory, control sequence, and the performance index

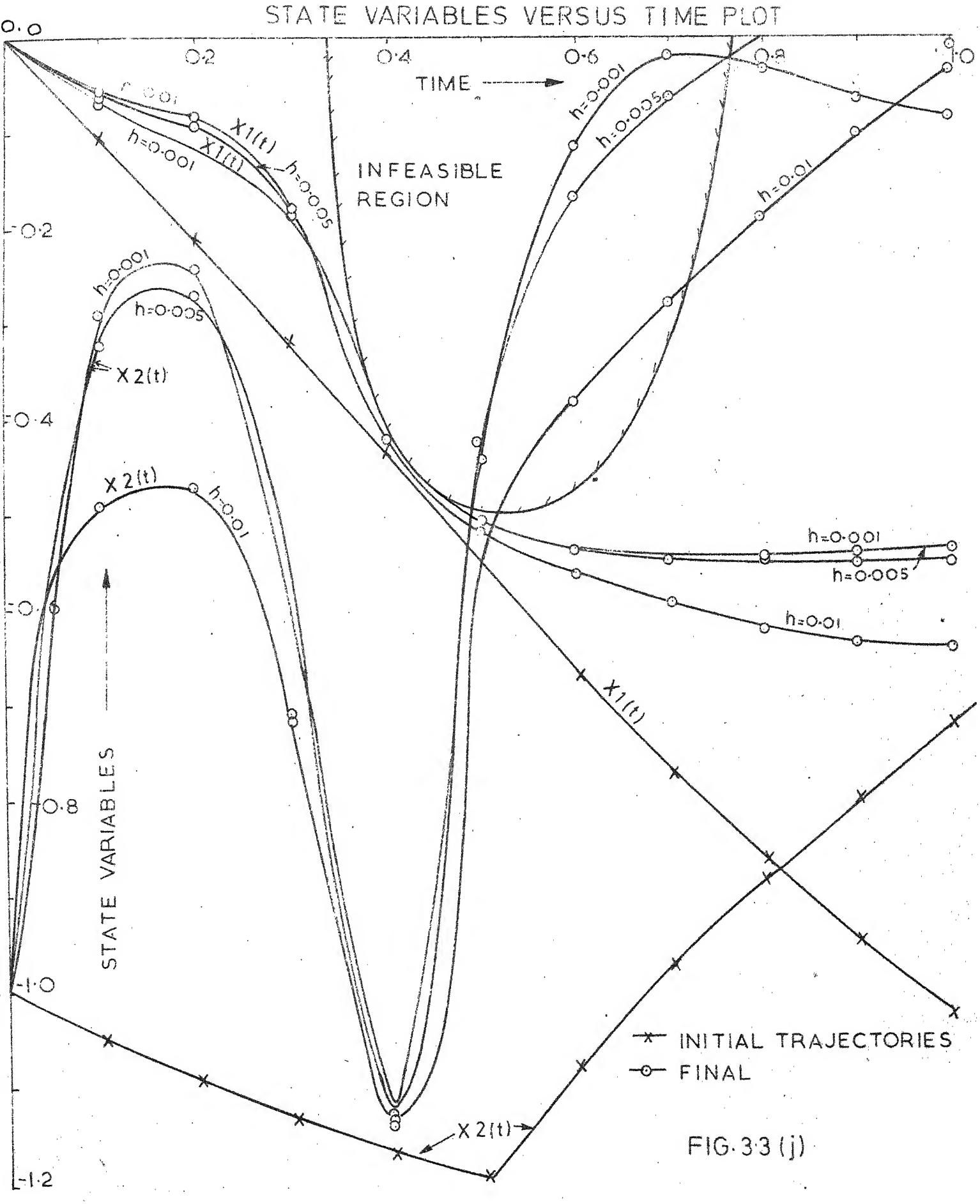

STATE VARIABLES VERSUS TIME PLOT
TIME
INFEASIBLE REGION
STATE VARIABLES
X1(t)
X2(t)
h=0.001
h=0.005
h=0.01
INITIAL TRAJECTORIES
FINAL
0.0
0.2
0.4
0.6
0.8
1.0
-0.2
-0.4
-0.6
-0.8
-1.0
-1.2

FIG. 3.3 (j)

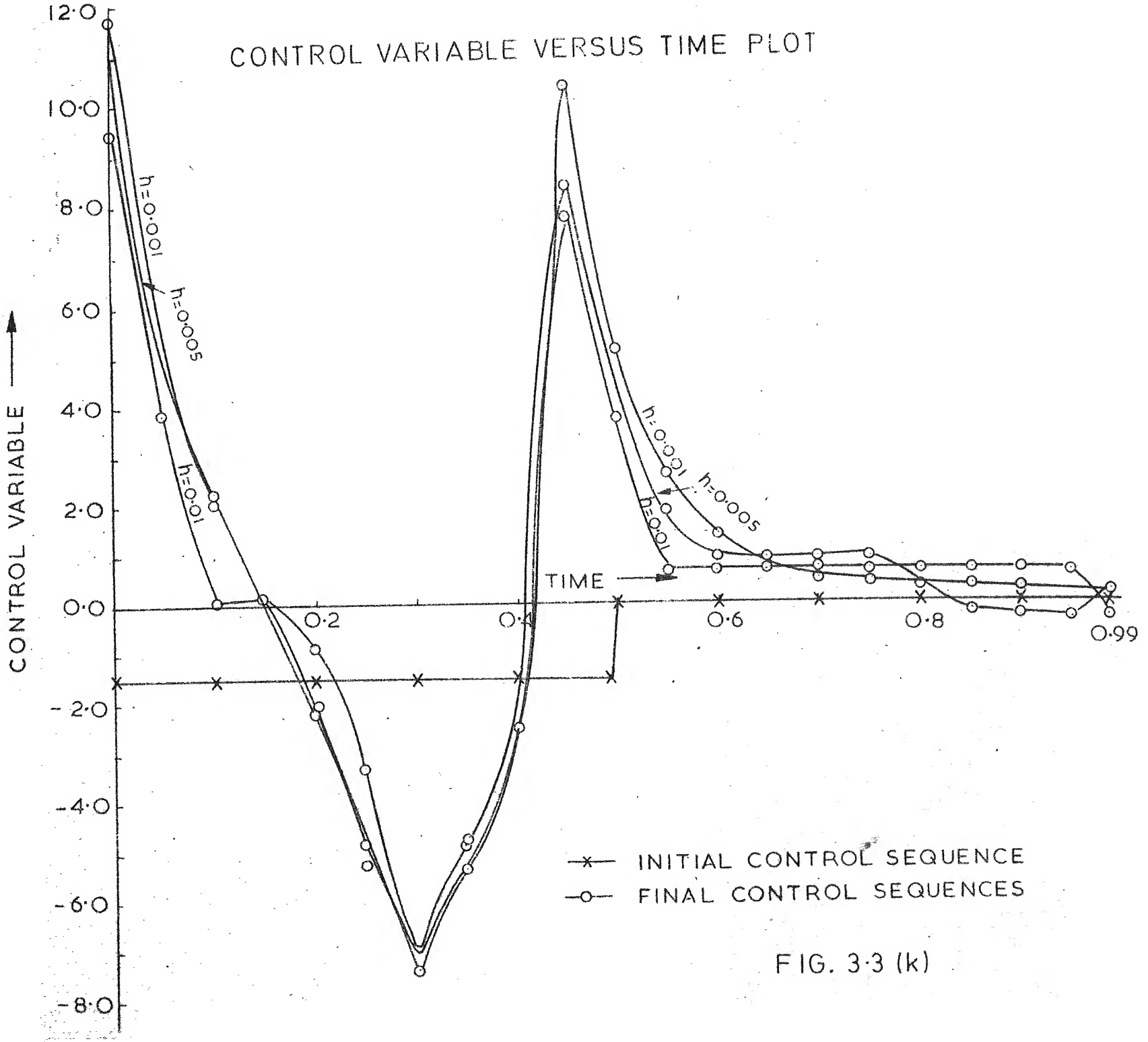

CONTROL VARIABLE VERSUS TIME PLOT
CONTROL VARIABLE
12·0
10·0
8·0
6·0
4·0
2·0
0·0
- 2·0
-4·0
-6·0
-8·0
h=0·001
h=0·005
h=0·01
h=0·001
h=0·005
h=0·01
TIME
0·2
0·4
0·6
0·8
0·99
INITIAL CONTROL SEQUENCE
FINAL CONTROL SEQUENCES

FIG. 3·3 (k)

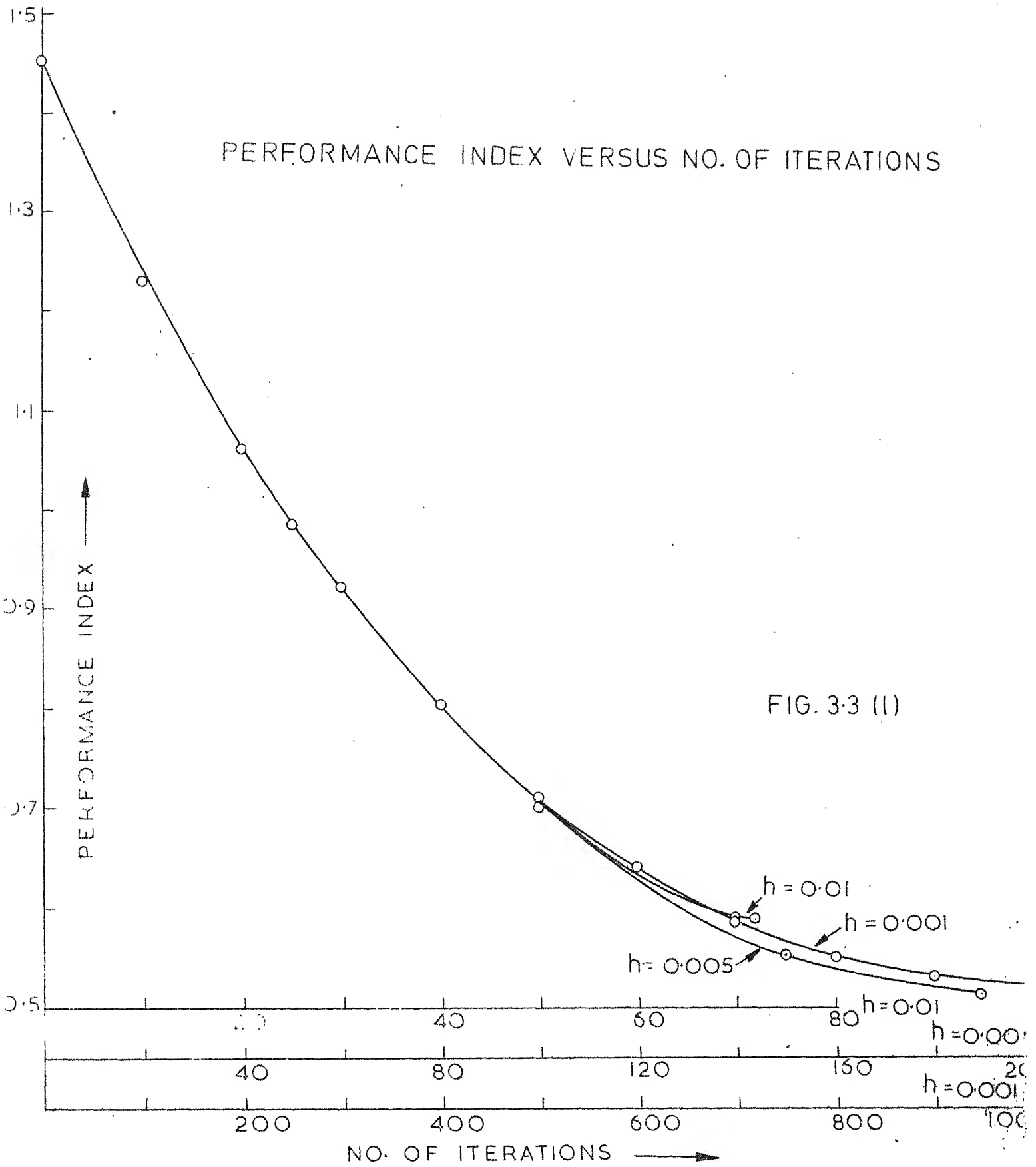

PERFORMANCE INDEX VERSUS NO. OF ITERATIONS
1·5
1·3
1·1
0·9
0·7
0·5
PERFORMANCE INDEX
h = 0·01
h = 0·001
h= 0·005
h=0·01
h=0·00
h=0·001
40
60
80
40
80
120
160
200
400
600
800
NO. OF ITERATIONS

FIG. 3·3 (I)

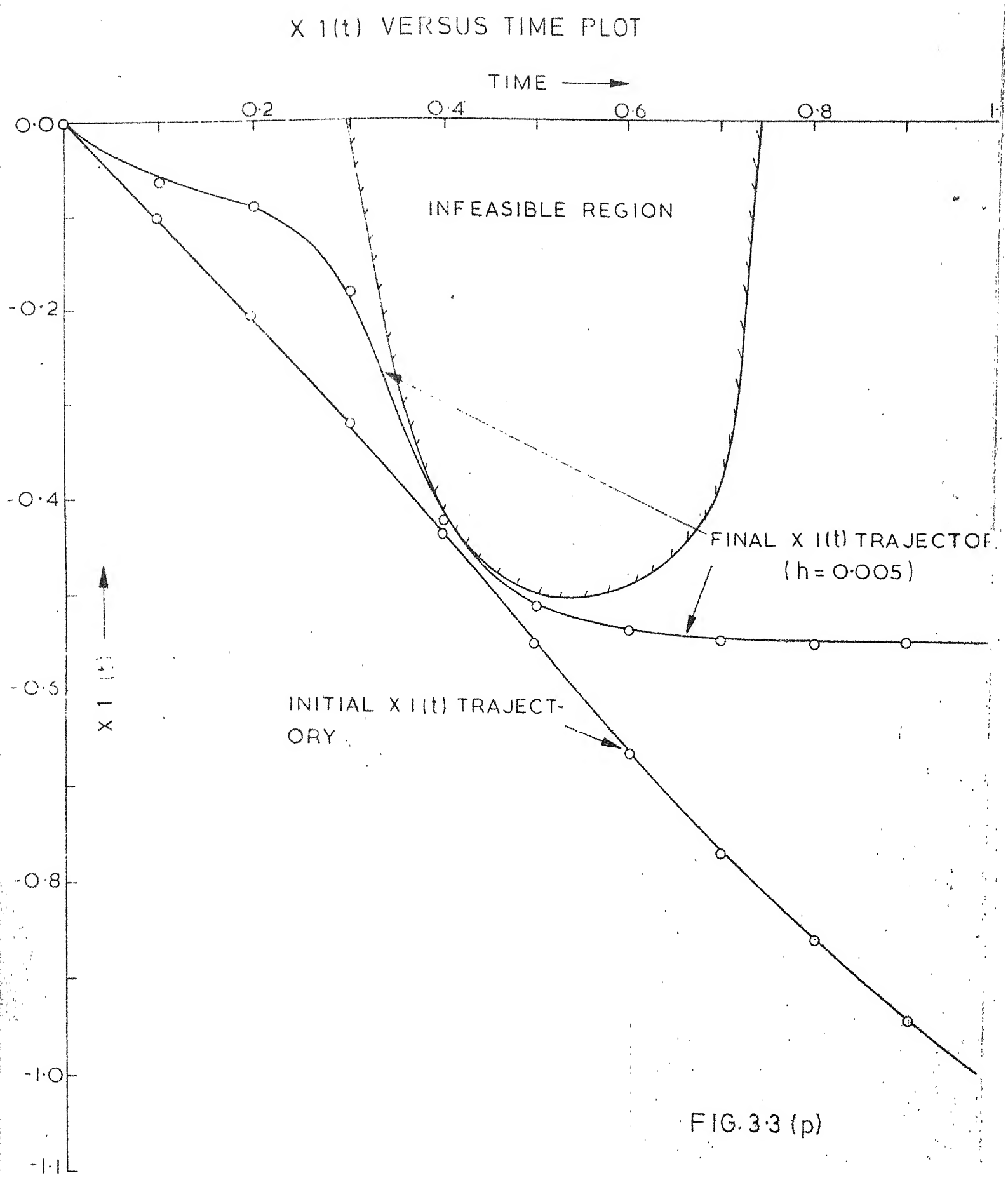

X 1(t) VERSUS TIME PLOT
TIME
0·2
0·4
0·6
0·8
1·
0·0
-0·2
-0·4
-0·6
-0·8
-1·0
-1·1
X 1 (t)
INFEASIBLE REGION
FINAL X 1(t) TRAJECTOR
(h = 0·005)
INITIAL X 1(t) TRAJECT-
ORY

FIG. 3·3 (p)

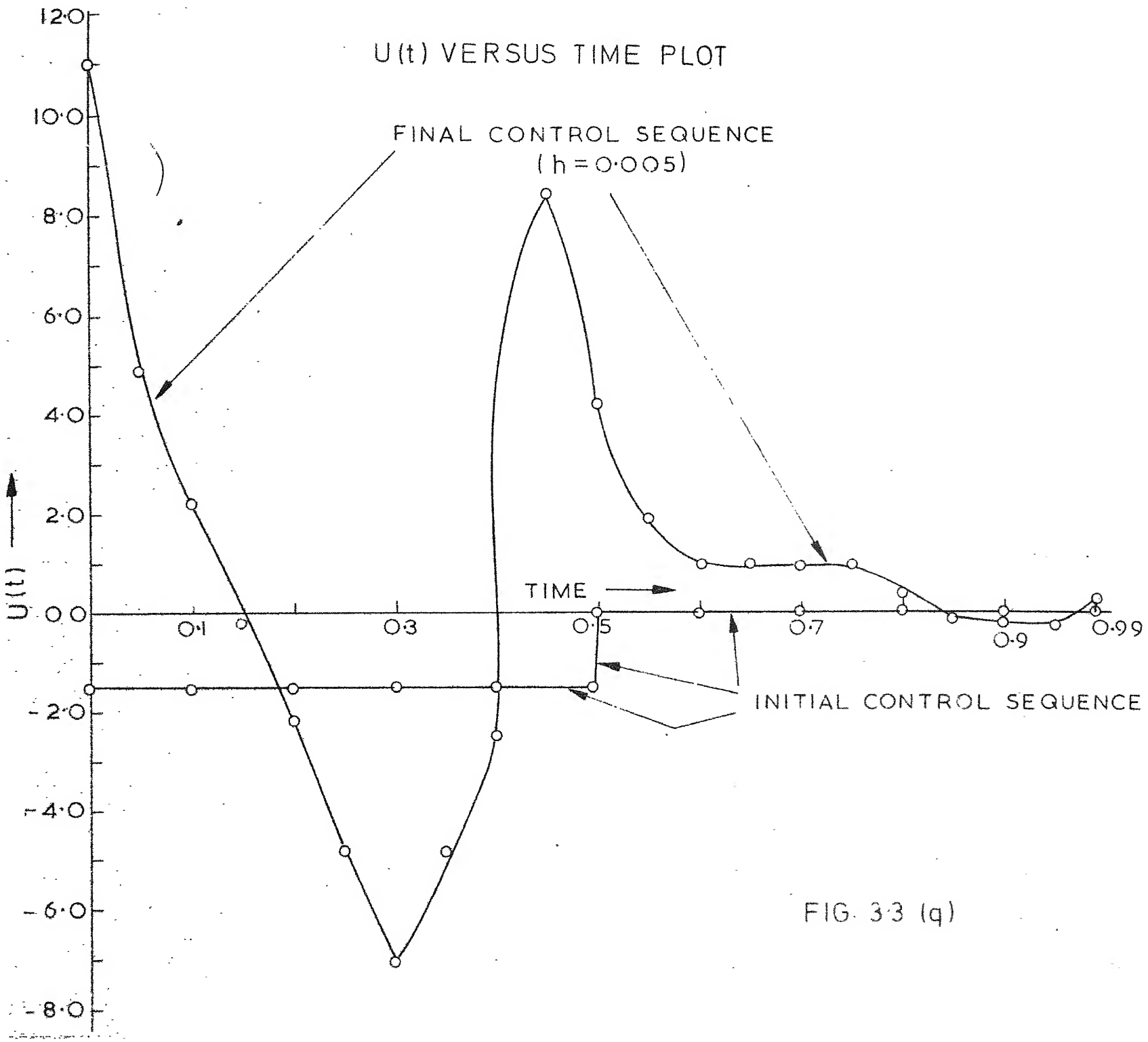

U(t) VERSUS TIME PLOT
FINAL CONTROL SEQUENCE
(h = 0·005)
INITIAL CONTROL SEQUENCE
TIME
U(t)
12·0
10·0
8·0
6·0
4·0
2·0
0·0
-2·0
-4·0
-6·0
-8·0
0·1
0·3
0·5
0·7
0·9
0·99

FIG. 3·3 (q)

J VERSUS K PLOT FOR H=0·005

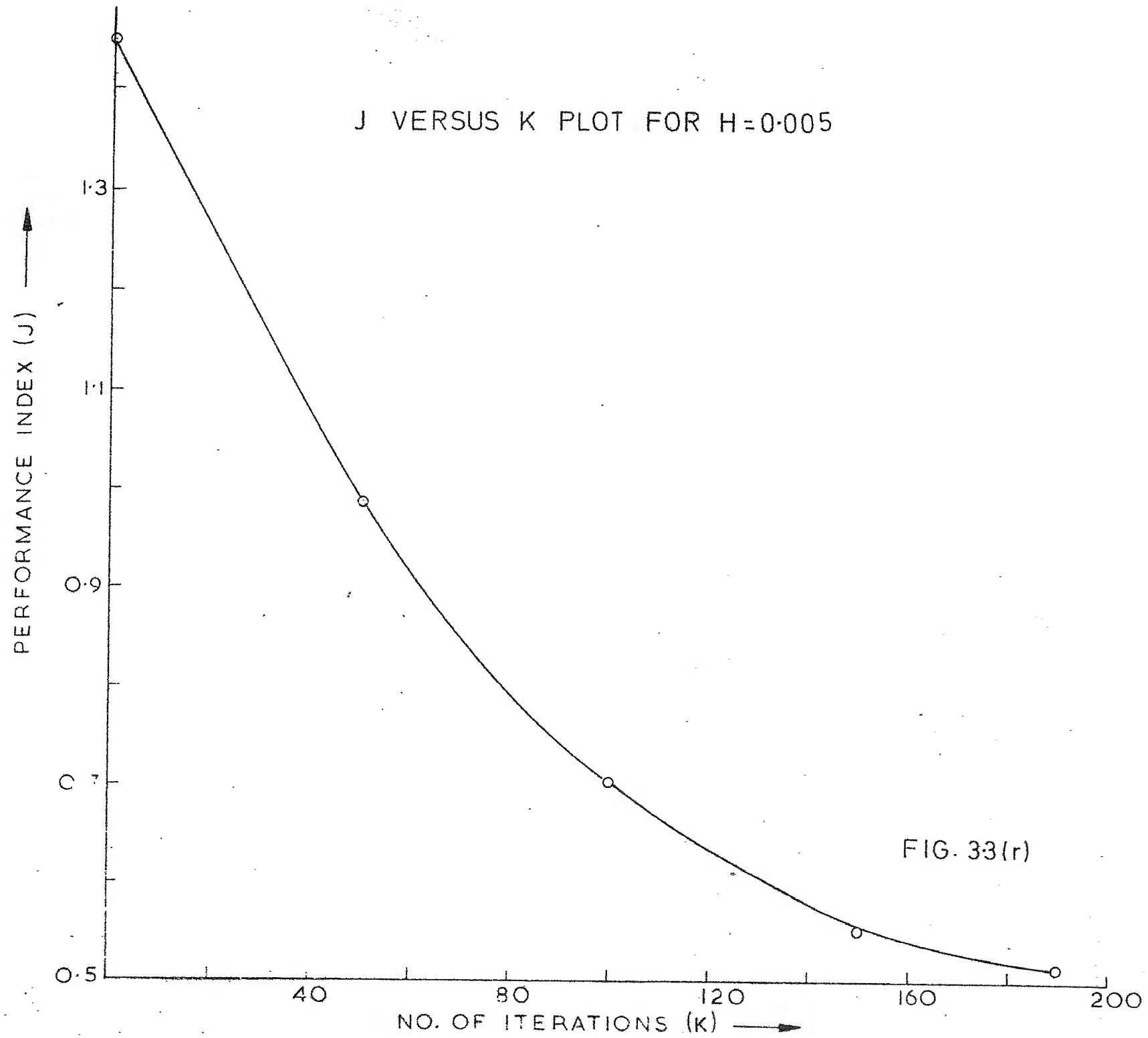


FIG. 3·3(r)

versus iteration number plots respectively, for $h = 0.005$, have been drawn.

3.3.5 Solution to Problem 4 with the MLV Algorithm modified as in Section 3.3.2

Double precision arithmetic was used in the computations and the results for the above modification to MLV, with $h = 0.005$ and $h = 0.001$, are presented in Table 3.3(f) below for comparison with the results for the original MLV as shown in Table 3.3(e).

Sl. No.	Initial perturbation step h_i	Final perturbation step h_f	Total No. of iterations K	Min.value of performance index J_{min}	Execution time on IBM 7044 (secs)
1	0.01	0.005	125	0.5271251	35.00
2	0.016	0.001	202	0.5008761	57.00

Table 3.3(f)

Comparing with Table 3.3(e), for $h = 0.005$ and $h = 0.001$, it is seen that the modification has significantly reduced the number of iterations required for convergence and the execution times for both cases. The value of J_{min} for $h = 0.001$ has also been reduced while the value of J_{min} for $h = 0.005$ has increased slightly.

3.3.6 Solution with the MLV Algorithm modified as in Section 3.2.2

Finally, Problem 4 was solved with the above modification using double precision arithmetic. Figures 3.3(m), (n) and (o) show the state trajectories, control sequences, and the performance index versus iteration number plots respectively, that were obtained. From the results tabulated in Table 3.3(g), the same conclusions as in Section 3.2.2 can be drawn.

Initial value of performance index = J_{ini}=1.454357

Sl. No.	Perturbation step h	No. of Iterations K	Min. value of performance index J_{min}	Execution time on IBM 7044 (secs)
1	0.01	58	0.6287411	27.00
22	0.005	74	0.7870553	37.00
3	0.001	94	0.8494794	83.00

Table 3.3(g)

From the results obtained in Sections 3.2.2, 3.3.3 and 3.3.6, the conclusion that can be drawn is that in terms of minimising the performance index, the MLV algorithm modified as in Section 3.2.2 does not give as good results as the original MLV algorithm.

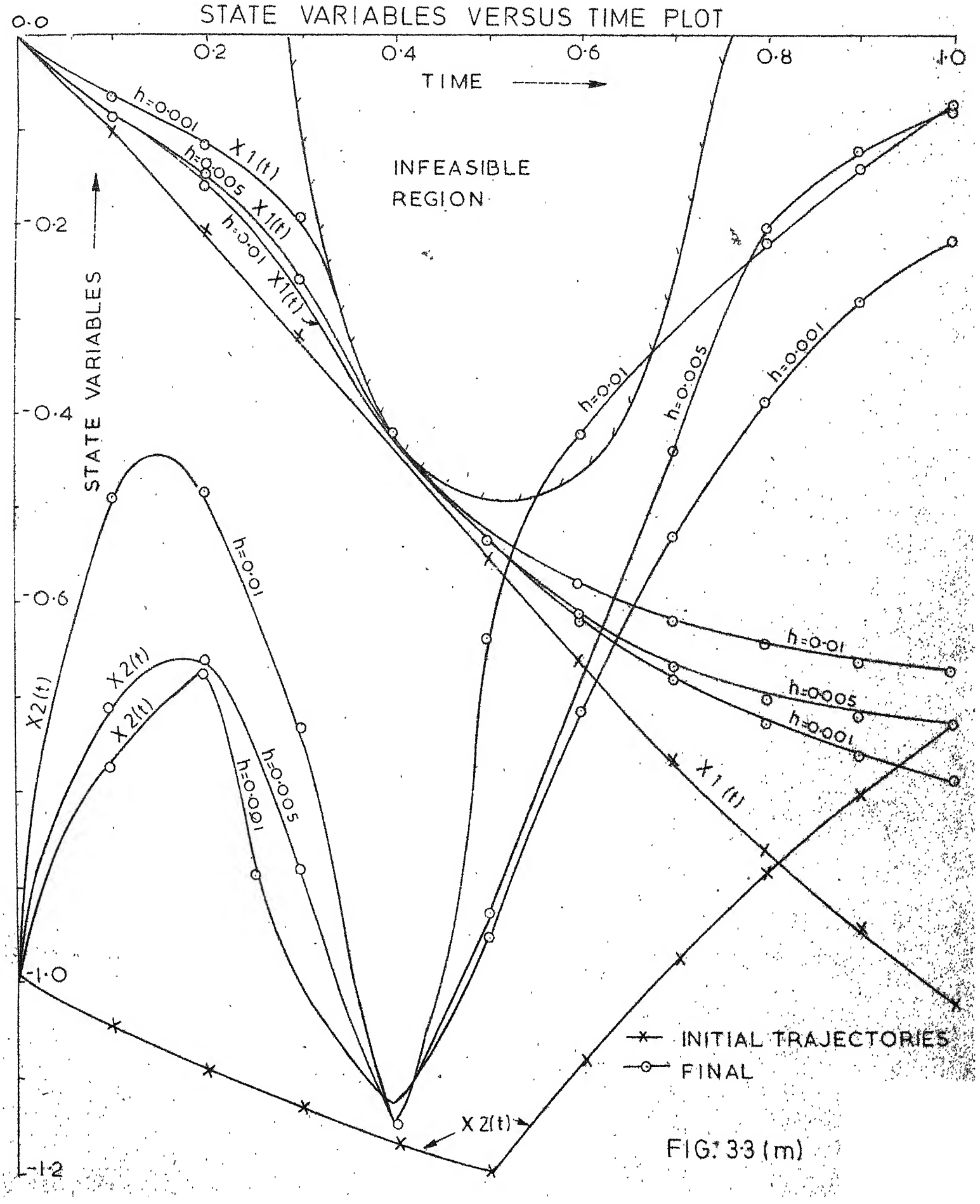

STATE VARIABLES VERSUS TIME PLOT
TIME
STATE VARIABLES
INFEASIBLE REGION
0·0
0·2
0·4
0·6
0·8
1·0
-0·2
-0·4
-0·6
-1·0
-1·2
h=0·001
h=0·005
h=0·01
X1(t)
X2(t)
INITIAL TRAJECTORIES
FINAL

FIG. 3·3 (m)

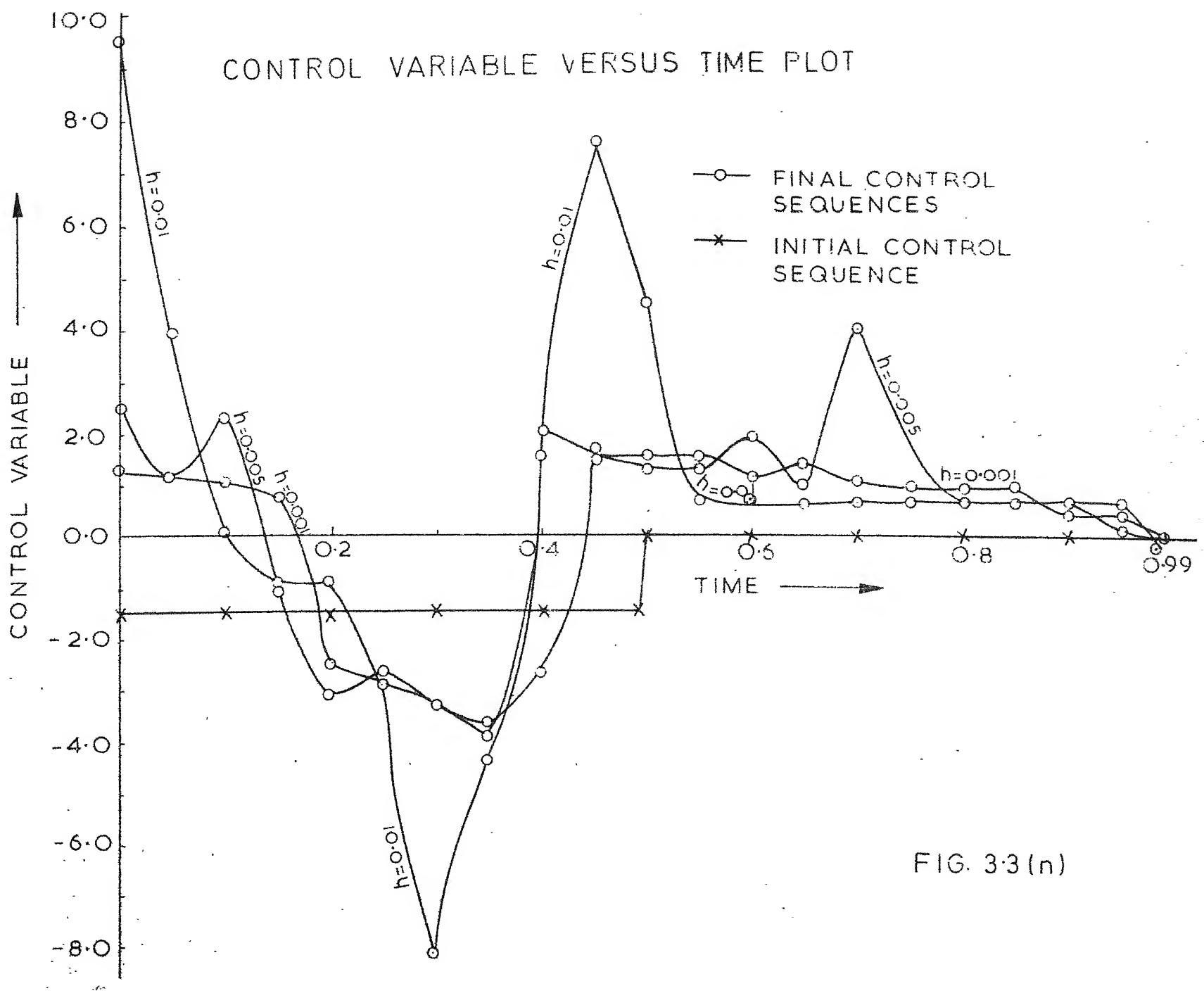

CONTROL VARIABLE VERSUS TIME PLOT
FINAL CONTROL SEQUENCES
INITIAL CONTROL SEQUENCE
CONTROL VARIABLE
TIME
10·0
8·0
6·0
4·0
2·0
0·0
-2·0
-4·0
-6·0
-8·0
0·2
0·4
0·6
0·8
0·99
h=0·01
h=0·005
h=0·001

FIG. 3·3 (n)

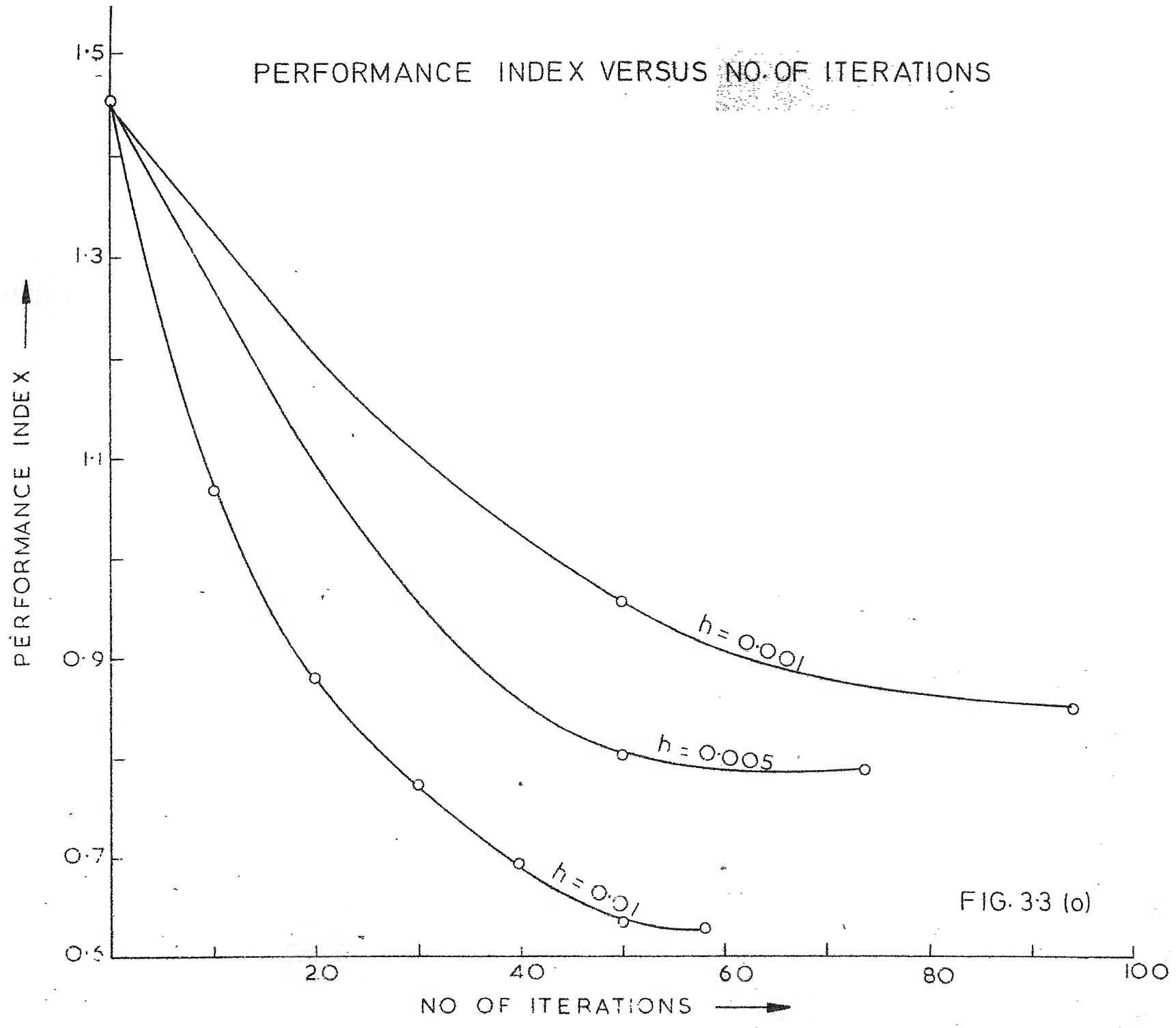

PERFORMANCE INDEX VERSUS NO. OF ITERATIONS
PERFORMANCE INDEX
1·5
1·3
1·1
0·9
0·7
0·5
h = 0·001
h = 0·005
h = 0·01
20
40
60
80
100
NO OF ITERATIONS

FIG. 3·3 (a)

In conclusion, it should be mentioned that in this chapter, the MLV algorithm and some modifications on it, were applied to solve some optimal control problems. As a result, some computational experience regarding this technique, which was hitherto lacking, has been gained. What is more, several important features of the MLV algorithm have emerged owing to its application to the problems solved. These traits of the algorithm can form a basis for its comparison with the more widely used techniques of optimisation.

CHAPTER 4

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

In the previous chapters, the Method of Local Variations (MLV) has been studied and the original MLV algorithm, along with some modifications on it, have been applied to solve several optimal control problems. The computational experience gained in this study has helped us to recognise several of its important features which can provide a basis for its comparison with other better known techniques of optimisation.

The Method of Local Variations has been described as a neighbourhood technique which utilizes a direct search procedure for optimisation. It is inherently simple and easily implementable, flexible and, thus, open to modifications, and efficient, in terms of memory requirement. Any state or control constraints stipulated in the problem ease the finding of a locally optimal solution. These constraints can be directly handled by the MLV technique. However, as indicated in Solution 1(a) to Problem 1 in Section 3.1, this method does not always guarantee convergence to a local optimum. Another drawback is that the MLV technique requires an initial feasible trajectory to start with, whose generation may prove difficult depending on the constraints present in the problem. Indeed this problem is encountered in many direct methods of optimization. It has also been shown in

Solution 1(b) to Problem 1 in Section 3.1 that the result of application of this method depends on the initial feasible trajectory chosen. However, this is not a unique feature of this method; the solution obtained by most of the iterative optimization techniques depends on the starting point for the iterations.

Among the modifications attempted on the original MLV algorithm, the one reported in Section 3.3.2 gave promising results in terms of significant reduction in the number of iterations required for convergence and computation time. There may well be other modifications which may yield still better results than those obtained in Section 3.3.2. For example, it would be interesting to see how the following MLV scheme works under the following conditions:

a) The time interval of the problem under consideration is first divided into N equal subintervals and the MLV algorithm modified as in Section 3.3.2 applied to obtain a solution.

b) With a starting point generated from the above solution (using an interpolation scheme), the same algorithm is applied to get another solution for a discretised-time model of the continuous system defined over 2N equal subintervals of the time interval of optimization.

This procedure is repeated, dividing the time interval into

smaller and smaller subintervals till a satisfactory convergence to a locally optimal solution is achieved. We note in passing that there may be computational advantage in considering partitioning of the time interval of optimization into judiciously chosen unequal subintervals.

In this thesis, only optimal control problems for linear systems have been solved by MLV. It is necessary to gain computational experience of MLV in its application to solve optimization problems defined for nonlinear systems. It would be worthwhile to investigate application of MLV to state equations linearized around the nominal trajectory and a simplified performance index defined only for the incremental state and control values. Since MLV is essentially a neighbourhood technique, we intuitively feel that such a scheme will work well, and there may be computational advantages to be gained. This linearizing idea can be found in differential dynamic programming technique [31].

Prakasa Rao et al. [38] have suggested a way of getting over the difficulty which arises out of the uncertainty in the MLV algorithm to find a locally optimal solution. When the initial nominal trajectory is far away from the optimal one, MLV can be applied to provide a faster transition from the current nominal trajectory to the next one. Once the current nominal trajectory enters a small neighbourhood

of the optimal one, we can switch over to the Incremental Dynamic Programming technique [30] which leads to the exact locally optimal trajectory.

Finally, we suggest a modification regarding the application of the MLV technique to solve multi-dimensional problems. The state trajectory, in this case, consists of a set of state-vector points at different instants tof time. According to the original algorithm, the first component of the state-vector, at a particular time instant is first perturbed by +h. If this is a 'success', the corresponding substitutions in state, control and incremental performance index values are made. If it is a 'failure', that component is then perturbed by -h. If it is a 'success', the necessary substitutions are made, otherwise, the old state, control and incremental performance index values are retained and the algorithm moves on to the second component of that state-vector, and so on, till all the components of that state-vector have been perturbed. Then the algorithm moves on to the state-vector at the next time instant and repeats the above process. This whole procedure is repeated till all the state-vector points, in the current nominal trajectory, have been subjected to the elementary operation.

The suggested modification consists in ordering the above search. For this, the past history in terms of

'success-failure' patterns, of the system state has to be remembered. If, say, the jth component of the state-vector at the ith time instant and kth iteration had been successfully perturbed by -h, then it would be "logical" to assume that, as MLV is a neighbourhood technique, the chances that a perturbation of -h would be 'successful' for the same component of the state-vector at the same time instant, in the (k+1)th iteration, are much higher than those for a perturbation of +h to that component. This could also be true for the jth component of the state-vector at the (i+1)th time instant and kth iteration. Thus, for this component of the state-vector at the (i+1)th time instant and kth iteration, and the ith time instant and (k+1)th iteration, we can directly give a perturbation of -h instead of first giving a perturbation of +h and then -h as in the original MLV algorithm. This should result in a significant reduction in execution time although this gain would be at the cost of an increase in memory requirement needed to "remember" the past 'success-failure' patterns.

From the above it is clear that many modifications of the basic MLV algorithm are possible, and there may be some which significantly reduce computational requirements. These are the possibilities that we see for extension of the work reported in this thesis.

REFERENCES

1. Athans, M.,'The Status of Optimal Control Theory and Applications for Deterministic Systems', IEEE Trans. on Automatic Control, Vol.AC-11, No.3, pp 580-596, July, 1966.
2. Kirk, D.E., Optimal Control Theory An Introduction, Prentice Hall, Electrical Engineering series.
3. Sage, A.P., Optimum Systems Control, Prentice Hall, Electrical Engineering Series.
4. Pierre, D.A., Optimisation Theory With Applications, John Wiley and Sons Inc.
5. Bellman, R., and Kalaba, R., Quasilinearisation and Nonlinear Boundary Value Problems, Elsevier Press, New York, 1965.
6. Bellman, R.E., Kagiwada, H.H., and Kalaba, R.E.,'Quasilinearisation, Boundary Value Problems and Linear Programming', The Rand Corporation Memo RM-4284-PR, September, 1964.
7. Bellman, R., Kalaba, R., and Sridhar, R.,'Adaptive Control via Quasilinearisation and Differential Approximation', The Rand Corporation, RM-3928-PR, November, 1963.
8. Kalaba, R., 'On Nonlinear Differential Equations, The Maximum Operation, and Monotone Convergence', Journal of Mathematics and Mechanics, Vol.8, pp 519-574, 1959.

9. McGill, R., and Kenneth, P., 'Solution of Variational Problems by Means of a Generalised Newton-Raphson Operator', AIAA Journal, Vol.2, pp 1761-1766, 1964.

10. Kopp, R.E., and McGill, R., 'Several Trajectory Optimisation Techniques; Part I: Discussion', A.V. Balakrishnan, and L.W. Neustadt, eds. Computing Methods in Optimisation Problems, Academic Press, N.Y., pp 65-89, 1964.

11. Moyer, H.G., and Pinkham, G., 'Several Trajectory Optimisation Techniques; Part II: Application', pp 91-105 of reference [10].

12. Kumar, K.S.P., and Sridhar, R.,'On the Identification of Control Systems by the Quasilinearisation Method', IEEE Trans. on Automatic Control, Vol. AC-9, No.2, pp 151-154, April, 1964.

13. Detchmendy, D.M., and Sridhar, R., 'On the Experimental Determination of the Dynamical Characteristics of Physical Systems,' Proc. of the National Electronics Conference, Vol.21, pp 575-580, 1965.

14. Sage, A.P., and Eisenberg, B.R., 'Experiments in Nonlinear and Nonstationary Systems Identification via Quasilinearisation and Differential Approximation', Proc. of the Joint Automatic Control Conference, pp 522-530, 1965.

15. Henrici, P., Discrete Variable Methods in Ordinary Differential Equations, John Wiley and Sons Inc., New York, 1962.

16. Sylvester, R.J., and Meyer, F., 'Two Point Boundary Value Problems by Quasilinearisation', Journal Society Industrial and Applied Mathematics, Vol.13, pp 586-602, June, 1965.

17. Sage, A.P., and Burt, R.W., 'Optimum Design and Error Analysis of Digital Integrators for Discrete System Simulation', American Federation of Information Processing Societies, Proc. F.J.C.C., Vol.27, Part I, pp 903-914, 1965.

18. Sage, A.P., and Smith, S.L., 'Real-Time Digital Simulation for Systems Control', IEEE Proc., Vol.54, No.12, pp 1802-1812, December, 1966.

19. Bellman, R., et al., 'Invariant Imbedding and Nonlinear Filtering Theory', Rand Corp., RM-4374-PR, December, 1964.

20. Detchmendy, D.M., and Sridhar, R., 'Sequential Estimation of State and Parameters in Noisy, Nonlinear Dynamical Systems', ASME Journal of Basic Engineering, Vol.88, Series D, No.2, pp 362-366, June, 1966.

21. Sage, A.P., and Masters, G.W.,'On-line Estimation of States and Parameters for Discrete, Nonlinear Dynamic Systems', Proc. of the National Electronics

Conference, Vol.22, pp 677-682, 1966.

22. Sage, A.P., and Masters, G.W.,' Identification and Modeling of States and Parameters of Nuclear Reactor Systems', IEEE Trans. on Nuclear Science, pp 279-285, February, 1967.

23. Sage, A.P., and Ellis, T.W., 'Sequential Suboptimal Adaptive Control of Nonlinear Systems', Proc, of the National Electronics Conference, Vol.22, pp 692-697, 1966.

24. Ellis, T.W., and Sage, A.P., 'Application of a Method of on-line Combined Estimation and Control,' Proc. Southwest IEEE Conference, April, 1968.

25. Lasdon, L.S., Mitter, S.K., and Waren, A.D., 'The Conjugate Gradient Method for Optimal Control Problems', IEEE Trans. on Automatic Control, Vol.AC-12, No.2, pp 132-138, April, 1967.

26. Bryson, A.E., and Ho, Y.C., Applied Optimal Control, Waltham, Mass: Blaisdell, 1969.

27. Bellman, R.E., and Dreyfus, S.E., Applied Dynamic Programming, Princeton, N.J: Princeton University Press, 1962.

28. Bellman, R.E., and Kalaba, R.E., Dynamic Programming and Modern Control Theory, New York: Academic Press, 1965.

29. Bellman, R.E., Dynamic Programming, Princeton, N.J: Princeton University Press, 1957.

30. Bernholtz, B., and Graham, L.J., 'Hydro-Thermal Economic Scheduling', Part I-Solution by Incremental Dynamic Programming, Trans. A.I.E.E., 1960, 79, (III), pp 921-932.

31. Jacobson, D.H., and Mayne, D.Q., Differential Dynamic Programming, American Elsevier Publishing Coy, Inc., New York, 1970.

32. Larson, R.E., 'Dynamic Programming with Reduced Computational Requirements', IEEE Trans. on Automatic Control, Vol.AC-10, pp 135-143, 1965.

33. Rosen, J.B., 'The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints', J. SIAM (1960), pp 181-217.

34. Rosen, J.B., 'Iterative Solution of Nonlinear Optimal Control Problems', J. SIAM Control Series A (1966), pp 223-244.

35. Rosen, J.B., 'Optimal Control and Convex Programming', Nonlinear Programming. J. Abadie, ed. New York : John Wiley and Sons, Inc., 1967.

36. Chernous'Ko, F.L., 'A Local Variation Method for Numerical Solution of Variational Problems', U.S.S.R. Computational Mathematics and Mathematical Physics, Vol.5, No.4, pp 234-242, 1965.

37. Krylov, L.A., and Chernous' Ko, F.L., 'Solution of Problems of Optimal Control by the Method of Local Variations', U.S.S.R. Computational Mathematics and Mathematical Physics, Vol.6, No.2, pp 12-31, 1966.

38. Prakasa Rao, K.S., Prabhu, S.S., and Aggarwal, R.P., 'Optimal Scheduling in Hydro-Thermal Power Systems by the Method of Local Variations', International Conference on Systems and Control, Aug.30-Sept.1, 1973.

39. Banichuk, N.V., Petrov, V.M., and Chernous' Ko, F.L., 'The Solution of Boundary Value Problems by the Method of Local Variations', U.S.S.R. Computational Mathematics and Mathematical Physics, Vol.6, No.6, pp 1-20, 1966.

41. Leondes, C.T., Ed., Advances in Control Systems, Vol.2, 1964-1966, New York: Academic.

42. Jacobson, D.H., and Lele, M.M., 'A Transformation Technique for Solving Optimal Control Problems with a State Variable Inequality Constraint', IEEE Trans. on Automatic Control, Vol.14, No.5, pp 457-464, October, 1969.

43. Mehra, R.K., and Davis, R.E., 'A Generalised Gradient Method for Optimal Control Problems with Inequality Constraints and Singular Arcs', IEEE Trans. on Automatic Control, Vol.17, No.1, pp 69-79, February, 1972.